TU WIEN Informatics

Unraveling Uncertainty Propagation in the Medical Visualization Pipeline

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Gabriel Häusle, BSc
Matrikelnummer 11808601

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dr.in Renata Georgia Raidou

Wien, 4. September 2025

Gabriel Häusle	Renata Georgia Raidou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Unraveling Uncertainty Propagation in the Medical Visualization Pipeline

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Informatics

by

Gabriel Häusle, BSc
Registration Number 11808601

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr.in Renata Georgia Raidou

Vienna, September 4, 2025

Gabriel Häusle	Renata Georgia Raidou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Gabriel Häusle, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 4. September 2025

Gabriel Häusle

Danksagung

Ich hatte das Glück, Associate Prof. Dr.in Renata Georgia Raidou als Betreuerin dieser Arbeit zu haben. Sie hatte immer Ideen, Zeit, Geduld und Optimismus, um auftretende Probleme zu bewältigen. Zusätzlich ermöglichte sie mir eine finanzielle Unterstützung meiner Arbeit:

Dieses Projekt wurde teilweise durch die *Career-Grant-Projekte* des Dekanats der Fakultät für Informatik der TU Wien finanziert (Projekttitel: *"Investigating Uncertainty Accumulation and Propagation through the Visualization Pipeline"*).

Ich danke außerdem meinen Mitbewohnern und Freunden, mit denen ich die Studienzeit in Wien sehr genossen habe. Abschließend danke ich meiner Familie und meiner Freundin, die mir das Studium ermöglicht und mich stets unterstützt haben.

Acknowledgements

I was fortunate to have Associate Prof. Dr.in Renata Georgia Raidou as my supervisor for this thesis. She always had ideas, time, patience, and optimism to help me tackle the problems that arose. Additionally, she made it possible for me to receive financial support for my work:

This project was partly funded through the *Career-Grant Projects* of the Dean's Office, Faculty of Informatics, TU Wien (Project Title: "*Investigating Uncertainty Accumulation and Propagation through the Visualization Pipeline*").

I also thank my roommates and friends, who made the time studying in Vienna a pleasure. Finally, I want to thank my family and my girlfriend for making my studies possible and for their ongoing support.

Kurzfassung

In der Datenvisualisierung bestehen die Quantifizierung, Sensibilisierung und Visualisierung von Unsicherheiten Herausforderungen, insbesondere in kritischen Domänen wie der Medizin. Medizinische Diagnosen und anschließende Behandlungen basieren immer auf menschlichen Entscheidungen, die ihrerseits aufgrund von Subjektivität oder Wahrnehmung anfällig für Unsicherheiten sind. Außerdem werden Entscheidungen häufig auf der Grundlage von Messungen oder Bildern getroffen, die selbst von Unsicherheiten beeinflusst sind, die durch Effekte wie Rauschen oder Auflösungsbeschränkungen verursacht werden. Daher ist der gesamte Diagnoseprozess in klinischen Umgebungen von verflochtenen Unsicherheiten geprägt, die sich akkumulieren und das Ergebnis einer Pipeline erheblich verändern können, mit potenziell nachteiligen Auswirkungen auf die Gesundheit der Patientinnen und Patienten, wenn diese Unsicherheiten nicht berücksichtigt werden.

Ziel der Arbeit ist es, zum Entwirren des komplexen Zusammenspiels von Unsicherheiten innerhalb der medizinischen Visualisierungspipeline beizutragen. Dies geschieht durch die Untersuchung der komplexen Phänomene der Unsicherheitsfortpflanzung in der medizinischen Visualisierungspipeline sowie durch die Extraktion und Analyse von Provenance-Informationen aus der Pipeline, eingebettet in ein interaktives Framework. Die Provenance-Informationen, die als vollständige Historie der Pipeline betrachtet werden können, ermöglichen den Vergleich der Ergebnisse der Unsicherheitsfortpflanzung verschiedener Pipeline-Zustände und liefern so Einblicke in das Verhalten von Unsicherheiten.

Um die konzeptionelle Wirksamkeit des Frameworks zu demonstrieren, werden aussagekräftige Anwendungsszenarien vorgestellt. Diese stellen einfache und komplexere Szenarien dar, um das Verhalten und die Auswirkungen verschiedener Arten von Parametern in der Pipeline zu analysieren. Außerdem werden Möglichkeiten aufgezeigt, wie Nutzende ihre Unsicherheit bezüglich bestimmter Bildregionen oder Parameter ausdrücken können und dadurch Einblicke in die Auswirkungen der spezifizierten Unsicherheiten gewinnen. Die Anwendungsszenarien betonen sowohl positive als auch negative Aspekte des Frameworks und geben den Nutzenden somit die Mittel, die zugrundeliegende Arbeit eigenständig zu bewerten.

Abstract

Quantifying, raising awareness, and visualizing uncertainty stand as challenges in data visualization, especially in critical application domains such as medicine. Medical diagnosis and following treatment are always based on human decision-making, which itself is prone to uncertainty due to subjectivity or perception. Furthermore, decisions are often taken by analyzing measurements or images, which themselves are affected by uncertainty, caused by effects such as noise or resolution limitations. Thus, the whole process of diagnosis in clinical environments is concerned with interwoven uncertainties that accumulate and may change a pipeline's result substantially, potentially with detrimental effects on the patient's health, if uncertainties are not considered.

This work aims to contribute to unraveling the complex interplay of uncertainties within the medical visualization pipeline. We do so by investigating the complex phenomena of uncertainty propagation in the medical visualization pipeline, in combination with extracting and analyzing provenance information from the pipeline encapsulated in an interactive framework. As a consequence, we utilize the provenance information, which can be seen as a complete history of the pipeline, to compare uncertainty propagation results of distinct pipeline states and thus gain insights into the behavior of uncertainty.

In order to demonstrate the conceptual effectiveness of the framework, meaningful usage scenarios are presented. Those lay out simple and more complex scenarios to analyze the behavior and impact of different sorts of parameters present in the pipeline. Furthermore, we present ways in which a user can express their uncertainty for certain image regions or parameters and thereby gain insights into the impact of the specified uncertainties. The usage scenarios emphasize both positive and negative aspects of the framework and thus provide users with the means to assess the underlying work independently.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Definition

Quantifying, raising awareness, and visualizing uncertainty stand as challenges in data visualization, especially in critical application domains such as medicine [GSWS21, RPHL14, HSB+22, LLPY07]. For instance, if a diagnosis or therapy relies on a visual representation of the corresponding patient data, the presence of uncertainty might disrupt the procedure or pose a threat to the patient's well-being [GSWS21]. Furthermore, uncertainty might obscure relevant information from the medical personnel and thus might adulterate diagnosis and further treatment.

Uncertainty is ingrained in every stage of the visualization pipeline [GSo06]—from data acquisition to filtering and mapping, as well as the rendering of the data by means of visual representations, and finally to interacting with these representations and deriving insights. While all those uncertainties undoubtedly impact the diagnosis in some way, the kind of influence varies a lot in shape, magnitude, and regularity. For instance, in image acquisition, the measurement precision might introduce some uncertainty; nevertheless, such uncertainty might be possible to estimate due to the resolution or other limiting factors of acquisition technology. Uncertainty on the other side, which is introduced due to different perceptions or further pursued to human interpretation, is possibly much harder to predict. Thus, extensive work has been put into uncertainty classification systems. One example is the system of Griethe et al. [GSo06], in which they define error, imprecision, accuracy, lineage, subjectivity, non-specificity, and noise as potential sources/types of uncertainty. Such uncertainty may have a significant impact on the individual steps of the visualization pipeline and, cumulatively, on its final outcomes. While a complete compensation for uncertainty might be impossible due to uncertainties that are hard to predict, it may be beneficial to raise awareness of uncertainty in diagnosis and treatment with the help of technology and training.

Previous work on uncertainty quantification, visualization, and assessment has laid the groundwork for understanding the intricacies of uncertainty within data visualization processes [GSWS21, Rai18, Wei22]. Researchers have developed methodologies to quantify, model, and visualize uncertainty at different stages of the pipeline [PRJ12], while various statistical techniques, simulation methods, and modeling approaches have also been employed to analyze the impact of uncertainty and its visualization [BWR24, SMH10]. Moreover, studies have investigated the impact of uncertainty on visualization outcomes, highlighting its influence on decision-making processes and the interpretation of visual representations [MMT+23]. In general, the overall term of *uncertainty quantification* is further split into *forward uncertainty quantification* and *backward uncertainty quantification*. Whereas the former is commonly referred to as *uncertainty propagation*, which is concerned with quantifying the uncertainty at the output of a model, while the input uncertainty is unknown [GSWS21]. Uncertainty propagation, sometimes also referred to as uncertainty analysis, has some similarities to parameter sensitivity analysis and thus is often confused or misused [SAB+19]. Parameter sensitivity analysis tries to quantify and model unknown behavior of a model by cycling through the input parameter space and thereby apportioning the influence to different input parameters [SAB+19], while uncertainty propagation on its own does not apportion anything and, as mentioned, is only concerned with the quantification of uncertainty arriving at the output of the model.

Finally, efforts have also targeted the design of provenance solutions that track the lineage of data to produce visualizations, enhancing transparency and reproducibility [BWD+19]. Although provenance is often complex and challenging to visualize, techniques have been researched that simplify provenance data to be better perceivable. Such techniques include, among other things, aggregation, graph merging, and shape and color variation [SLSG16, KOF+16, AKM+18]. Developed standards to display provenance data and tools also exist [HG15, GM13], but whether a provenance visualization is useful depends heavily on the context.

1.2 Aim of this Work and Contribution

Despite the previously discussed advancements, the state of the art in uncertainty quantification and visualization lacks standardized methodologies for assessing the accumulation and propagation of uncertainties across multi-step processes, such as the visualization pipeline. Additionally, there is a need for provenance solutions that track uncertainty propagation and accumulation effectively through the visualization pipeline. This thesis intends to cover these two gaps in the literature by means of an interactive visual analysis framework.

Our project aims to unravel the complex interplay of uncertainties resulting from each step of the visualization pipeline. Specifically, we are interested in investigating *uncertainty propagation*, i.e., the process by which uncertainty increases or accumulates as data progresses through the steps of a visualization pipeline [Rai18, GSWS21]. We also clarify the difference between the general terms of parameter sensitivity analysis and

uncertainty quantification and explain their respective sub-terms. After the comparison, it is visible that our project utilizes a bit of parameter sensitivity analysis, while mainly being about uncertainty propagation. Those concepts are essential for understanding, managing, and minimizing—when possible; otherwise raising awareness about—the effect of uncertainties through the visualization pipeline. Yet, the complex nature of uncertainty has not allowed for a comprehensive investigation of its accumulation and propagation mechanisms [Rai18].

Our methodology aims to integrate statistical techniques, simulations, and modeling approaches within a framework for the quantification, exploration, analysis, and assessment of the impact of uncertainty on visualization pipeline outcomes. By subjecting our framework to diverse scenarios from the domain of medical visualization, we aim to reveal how different uncertainties shape visualization outcomes and influence the target user's decision-making processes. We also aim to integrate *provenance solutions* [BWD+19, MCER25] that meticulously track the lineage of uncertainties through the visualization pipeline to support transparency, reproducibility, and accountability—all being essential pillars of reliability and trustworthiness in visualization.

The central questions of this project can be summarized as follows:

RQ1. How does uncertainty propagate through the steps of the (medical) visualization pipeline?

RQ2. How can we integrate provenance solutions into the (medical) visualization pipeline to track and manage uncertainty effectively?

We hypothesize that uncertainties introduced at the different visualization pipeline steps contribute to the final outcome of the pipeline in a complex, but quantifiable, predictable, and modelable manner. Furthermore, we hypothesize that implementing provenance solutions to systematically assess the contribution of these individual uncertainties to the final outcome of the pipeline will unveil their respective impact. This impact, if transferred into real-life scenarios, results in less error-prone diagnoses and more uncertainty-aware actions by medical personnel.

We contribute by developing an interactive framework that couples pipeline building and provenance to present a proof of concept on how uncertainty quantifications on distinct pipeline states can be utilized to empower detailed uncertainty analysis. From the gained insights, medical personnel can benefit in various ways, such as analyzing the behavior of uncertainties, analyzing the influence of modules or pipeline configurations on uncertainty, and thus improving treatment and diagnosis accordingly.

1.3 Structure

The thesis is structured in the following way: First, Chapter 2 offers a detailed view into the related work of the topics covered by the thesis. Moreover, Chapter 3 presents and

explains the methodology of the developed framework. Chapter 4 extends the methodology with some corresponding implementation details and obstacles. In Chapter 5, the results are laid out and discussed critically. The last Chapter 6 wraps up the thesis by concluding and mentioning future work.

CHAPTER 2

Background & Related Work

2.1 Visualization Pipeline

2.1.1 General Visualization Pipeline

Moreland et al. [Mor13] describe the visualization pipeline as a directed data-flow network composed of executable modules that implement a computational process together. These modules are connected in a directed graph and are classified into three categories:

- *Source:* generates data and provides it at the output.
- *Filter:* transforms the data from the inputs and provides it at the outputs.
- *Sink:* handles the input data in various ways without providing any output for the following modules.

The purpose of a pipeline is to connect such modules meaningfully, thereby defining a high-level algorithm that governs the flow of data through multiple transformations [Mor13]. Haber et al. [Hab90] define three stages for visualization pipelines as *Data Enrichment/Enhancement*, *Visualization Mapping*, and *Rendering*. Brodlie et al. [BAL12] extend that definition into *Data*, *Filter*, *Map*, and *Render*, which can be seen in Figure 2.1. The *Data* stage involves the acquisition of the raw or simulated data. Next, the *Filter* stage creates an empirical model, for example, by constructing implicit surfaces or computing statistical summaries. The *Map* stage computes the geometrical elements and maps the data to visualization attributes like color, opacity, and texture. Lastly, *Rendering* involves the visualization of scene objects into a displayable image.

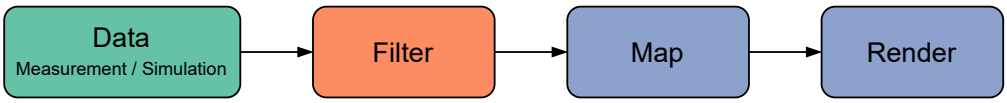


Figure 2.1: Overview of the visualization pipeline with its stages. Figure adapted from Brodlie et al. [BAL12] and Haber et al. [Hab90].

2.1.2 Medical Visualization Pipeline

The medical visualization pipeline is used in medical imaging for diagnosis, treatment planning, intraoperative support, and education [PB14a]. Diagnosis describes the decision about the type of disease that is present. Visualizing and thus characterizing the pathology provides valuable information for clinicians. Treatment planning refers to developing a therapy plan, where medical imaging allows for precise localization and examination of the affected tissue. During surgery, intraoperative support provides real-time imaging to provide the medical team with updated visual guidance. In medical education, high-quality visualizations of the human body offer realistic representations of anatomy and pathology, enhancing both learning and understanding [GSWS21].

The medical visualization pipeline, see Figure 2.2, is typically divided into three stages: *Acquisition*, *Transformation*, and *Visualization*. Each stage will be explained in the following [GSWS21].

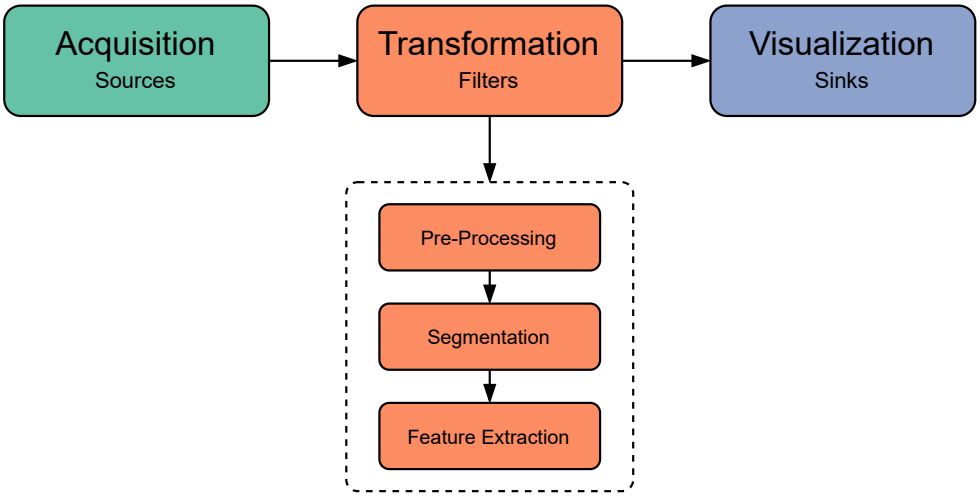


Figure 2.2: Overview of the medical visualization pipeline with its stages and the module types used in the respective stages. The figure merges the abstractions of Gillmann et al. [GSWS21] and Moreland et al. [Mor13].

Acquisition

The acquisition stage is where data enters the pipeline. Modules at this stage act as *sources*, thus the data collected at this stage often stems from medical imaging modalities. Such modalities include X-rays, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and Ultrasound [PB14c, GSWS21, Mor13].

Transformation

Filters in this stage transform the data emitted from the acquisition stage and pass it on to the visualization stage. The transformation stage can be split into three sub-stages: pre-processing, segmentation, and feature extraction [GSWS21].

The raw data collected at the acquisition stage can include artifacts, inhomogeneities, and noise. One goal of pre-processing is to balance between increasing the signal-to-noise ratio (SNR), which simplifies further image analysis, while maintaining all relevant small structures. Another goal of pre-processing is to improve the visual quality of the image, for example, by increasing contrast. Possible techniques include image enhancement, noise reduction, color correction, and edge detection. Medical applications often require multiple datasets from different modalities, which must be aligned. This process—known as registration—is also a pre-processing sub-stage [PB14d, GSWS21, Mor13].

The segmentation stage focuses on extracting the desired information from the pre-processed data. Segmentations can range from whole organs to tumors or blood vessels. Various segmentation strategies exist, such as manual segmentation, threshold-based region growing, watershed segmentation, or livewire methods [PB14d, GSWS21].

Feature extraction is about computing information from the processed dataset. Such information includes surface meshes, quantitative measurements, or trajectory paths [GSWS21, PB14b].

Visualization

In the final stage, modules that act as *sinks* visualize the data for the user, which can be in 2D, 3D, or even higher-dimensional (e.g., when including time). While 2D visualizations are relatively straightforward, 3D visualization techniques are more complex. The two main 3D visualization approaches are surface rendering and direct volume rendering. Surface rendering computes a mesh around an object of interest, which is then rendered; in this way, only the hull of the object is visible. Direct volume rendering, on the other hand, projects rays through the entire volume onto an image or camera plane, allowing the internal structures of the volume to be visualized as well [GSWS21, PB14e, PB14f, Mor13].

2.2 Uncertainty

2.2.1 Definitions

Uncertainty is a term used in many scientific disciplines; consequently, finding a single definition that captures all of its facets is challenging. Hunter et al. [HG93] define uncertainty as the "degree to which the lack of knowledge about the amount of error is responsible for hesitancy in accepting results and observations without caution." Thomson et al. [THM+05] generalize several definitions to: "definitions of uncertainty imply that there is imperfection in the users' knowledge about a dataset, process, or result."

Kiureghian et al. [KD09] discuss a common approach to classify uncertainty into two types: aleatoric and epistemic. The Latin root *alea*, meaning "the roll of a die," refers to uncertainty that is presumed to be the intrinsic randomness of a phenomenon. In contrast, the Greek root *episteme*, meaning "knowledge," denotes uncertainty that is presumed to be caused by a lack of knowledge. Potter et al. [PRJ12] offer a similar view, explaining that aleatoric uncertainty stems from fundamental randomness that cannot be reduced, whereas epistemic uncertainty arises due to incomplete knowledge that, in principle, could be taken into account. However, Kiureghian et al. [KD09] suggest that the categorization into aleatoric or epistemic can vary in different scenarios because it is subjective and context-dependent. They even raise the question of whether any aleatoric uncertainty exists at all, asking if at some point knowledge advances so far that all aleatoric uncertainty turns epistemic. Aside from this philosophical debate, in current practice, it makes sense to categorize some uncertainties as aleatoric, to mark which uncertainties are hard or impossible to reduce with current knowledge.

Brodlie et al. [BAL12] emphasize the distinction between *visualization of uncertainty* and *uncertainty of visualization*. The former term refers to uncertainty that already exists in the collected data or is introduced during the acquisition process, whereas the latter term summarizes uncertainty added by the various transformations within the visualization pipeline. *Visualization of uncertainty* corresponds to the aforementioned pipeline stages of data/acquisition ■, while *uncertainty of visualization* filter/transformation ■ and map, render/visualization ■. With those high-level definitions and pipeline stages mapped out, we can now examine more granular classification schemes.

2.2.2 Classifications

To enhance uncertainty awareness in applications, it is helpful to classify every type of uncertainty into a more detailed system. Multiple such classification concepts for uncertainty exist. Griethe et al. [GSo06] merge concepts from multiple previous works into the following taxonomy:

- *Error:* outlier or deviation from a true value.
- *Imprecision:* resolution of a value compared to the needed resolution.

- *Accuracy:* size of the interval a value lies in.
- *Lineage:* source of the data.
- *Subjectivity:* degree of subjective influence in the data.
- *Non Specificity:* lack of distinction for objects (e.g., when the category but not the sub-category of an object is known).
- *Noise:* background noise.

Thomson et al. [THM+05] propose a classification for geopolitically referenced information that defines the following uncertainty types:

- *Accuracy/error:* difference between observation and reality.
- *Precision:* exactness of measurement.
- *Completeness:* extent to which info is comprehensive.
- *Consistency:* extent to which info components agree.
- *Lineage:* conduit through which info passed.
- *Currency/timing:* temporal gaps between occurrence, info collection & use.
- *Credibility:* assessment of info source.
- *Subjectivity:* amount of interpretation or judgment included.
- *Interrelatedness:* source independence from other information.

Pang et al. [PWL97a] take a more simplistic approach and only use fewer types:

- *Statistical:* either given by estimates or actual data distribution.
- *Error:* signed or absolute error value.
- *Range:* a range in which the data must exist, but cannot be expressed either as an error or statistical uncertainty.
- *Judgment:* uncertainty in the process of synthesizing all relevant information of a process.

The aforementioned classification systems differ considerably yet share some fundamental overlaps, likely because they target different domains. To address this, Skeels et al. [SLSR10] analyzed multiple classification systems across multiple fields and interviewed 18 domain experts about the uncertainty in their respective data. They then proposed a more generalizable classification system as shown in Figure 2.3 and introduced levels of uncertainty to capture higher and lower forms of uncertainty. The resulting uncertainty types are the following:

- *Measurement Precision, Level I*: refers to uncertainty in "any variation, imperfection or theoretical precision limitations in measurement techniques that produce quantitative data."
- *Completeness, Level II*: includes uncertainty in sampling, missing values, and uncertainty in aggregation.
- *Inference, Level III*: depicts uncertainty in: modeling, prediction, and extrapolation. "It is how data are infused with meaning and transformed into decisions."

In addition to those three general types, another two types—*Credibility and Disagreement*—are proposed that span across Level I-III, and therefore can occur at each level:

- *Credibility*: refers to uncertainty about the credibility or trustworthiness of a data source, a subjective judgment of whether a source is credible, and past event credibility, questioning events.
- *Disagreement*: refers to disagreeing values, datasets, or conclusions. *Disagreement* can result in *Credibility* uncertainty.

Skeels et al. [SLSR10] highlight two takeaways: their classification may not cover all domain-specific uncertainties, and it does not target uncertainty that the visualization itself would introduce. Although having a cross-disciplinary classification for uncertainty is beneficial, the choice of which system to use remains application-dependent. With different classification approaches discussed, we can now look into sources of uncertainty.

All of the mentioned classification systems are partially concerned about the origin and reason for uncertainty. For example, they all include a category that is concerned with uncertainty due to subjectivity. Most comprise even more such categories. As this work comprises a general approach to uncertainty, we do not care about the reason or origin of uncertainty, and, thus, we do not follow any of the mentioned classification systems. Nevertheless, all classifications are consistent and have their raison d'être.

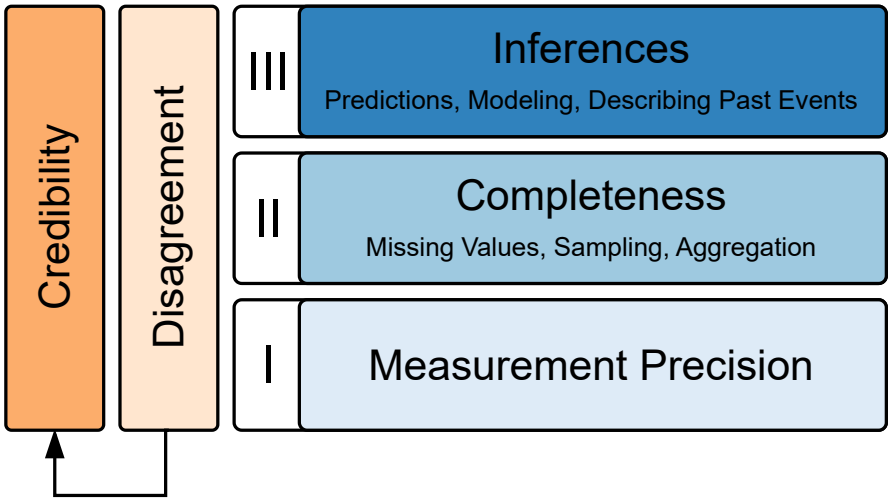


Figure 2.3: Uncertainty classification system adapted from Skeels et al. [SLSR10]. There are three uncertainty level categories: Measurement Precision, Completeness, and Inferences. The two other uncertainty categories, Credibility and Disagreement, span across all three levels. Moreover, disagreement may lead to credibility uncertainty.

2.2.3 Uncertainty Sources in the Medical Visualization Pipeline

Sources of uncertainty may occur at each step in the visualization pipeline [GSo06, BAL12, HSB+22]. Gillmann et al. [GSWS21] present three uncertainty categories for each step of the pipeline. The following subsections comprise those categories, and in addition, we refer to Ristovski et al. [RPHL14] for some specific uncertainty sources in medical applications.

Acquisition

Positional uncertainty (aleatoric)

Uncertainty in the spatial placement of each captured image. This information is crucial for possible registration operations further in the pipeline.

Pixel/voxel uncertainty (aleatoric)

Influences in the signal-acquisition process cause intensities to deviate unpredictably from their true values.

Incompleteness of data (aleatoric)

Loss of information when converting an analog signal into a discrete representation, which, out of necessity, has a limited resolution, leading to uncertainty.

Acquisition stage examples include:

- *Image reconstruction*: uncertainty arising from the assumptions made during the computational process of transforming raw sensor data from imaging modalities into images.
- *Noise*: refers to random variations in the scanning technique that affect the measured signal used for reconstruction.
- *Scanning artifacts*: uncertainty that occurs due to scanning technique-related imaging artifacts such as *bias field (MRI)* or *shadows (CT)*.
- *Partial volume effect*: introduces uncertainty when a voxel or pixel contains multiple tissue types, thus its intensity is ambiguous.
- *Patient motion*: uncertainty that arises due to the motion of the patient and organs during image acquisition.

Transformation

Model inaccuracy (epistemic)

Uncertainty is caused by simplifications in model design due to the inability to model real physical behavior perfectly.

Model imprecision (epistemic)

Uncertainty is introduced by numerical approximations performed during the transformation stage.

Parameter/boundary condition uncertainty (aleatoric and epistemic)

Uncertainty in the selection of parameters or boundary values that are typically determined experimentally, and thus without a definitive ground truth.

Transformation stage examples include:

- *Registration*: refers to the alignment of datasets of multiple modalities or the alignment of images over time, which both introduce uncertainty due to insufficient positional information.
- *Segmentation*: algorithmic or manual labeling of voxels to structures carries uncertainty from model assumptions and selected parameters.
- *Morphological operations*: refer to image processing operations that change the image itself and therefore uncertainty estimates, but do not introduce any uncertainty themselves.

- *Diffusion tensor imaging (DTI)*: refers to a MRI modality that measures water diffusion in tissue by fitting a diffusion-tensor model to the acquired data under defined assumptions. This enables estimation of neural fiber orientation and structure with uncertainty arising from those model assumptions.
- *Simulation parameters*: while the range of simulation parameters is often known, uncertainty arises due to their unknown exact value.

Visualization

Rasterization uncertainty (epistemic)

Uncertainty is caused by a loss of detail when projecting a high-resolution visualization model onto a lower-resolution screen.

Perceptual and cognitive uncertainty (aleatoric and epistemic)

Uncertainty resulting from variations in human perception and cognitive interpretation of the same visualization causes humans to draw different conclusions.

Decision-making bias (aleatoric and epistemic)

Uncertainty is induced by personal decision-making biases, which can further affect an interaction with a visualization.

Visualization stage examples include:

- *Interpolation*: is needed to compute intensities at positions lying between data samples. Different interpolation models carry their assumptions, and thus, their uncertainties induce uncertainty based on their assumptions.
- *DVR reconstruction*: projects volumetric data onto a 2D screen by numerically approximating the rendering integral, thereby introducing discretization uncertainty.
- *Human interpretation*: uncertainty arises from differences in perception and cognition, influenced by factors like color, interaction, and prior experience.

Considering Gillmann's classification alongside Ristovski's examples, one might expect a straightforward mapping. However, Ristovski's examples remain quite broad, covering diverse methods and techniques, so that most cases span multiple uncertainty categories. For a precise mapping, the medical visualization pipeline should be decomposed into atomic uncertainty-source steps. This would eliminate classification ambiguity but, conversely, produce an extensive list of atomic uncertainty sources. Take image reconstruction as an example: although it belongs to the acquisition stage, it invokes transformation-stage categories. *Model inaccuracy* applies because of the algorithms' underlying assumptions [PA23]. Whereas *model imprecision* applies because of inherent numerical limitations. Moreover, if the algorithm depends on tunable settings, *parameter/boundary-condition* uncertainty applies as well. Hence, forcing each example into a single category obscures more than it clarifies.

2.2.4 Uncertainty Quantification and Propagation

Uncertainty quantification is a term that describes *forward uncertainty quantification* and *backward uncertainty quantification*. The latter intends to quantify the difference between a mathematical model and an experiment. Whereas the former, typically referred to as *uncertainty propagation*, aims to quantify the propagation and accumulation of uncertainty through a model [GSWS21].

Zhang et al. [Zha21] define uncertainty propagation as follows. Consider a random variable X that is distributed according to the probability density function $p_X(x)$. Another random variable at the output Y is defined by the transformation of the model as $Y = \mathcal{M}(X)$. Uncertainty propagation aims to quantify the statistics of Y e.g. the probability density function $p_Y(y)$ with the expectation,

$$\mathbb{E}[\mathcal{M}] = \int_{-\infty}^{\infty} \mathcal{M}(x) p_X(x)\, dx \tag{2.1}$$

and the variance,

$$\mathbb{V}[\mathcal{M}] = \mathbb{E}[\mathcal{M}^2] - \mathbb{E}[\mathcal{M}]^2. \tag{2.2}$$

An illustration of the just explained can be seen in Figure 2.4. $p_Y(y)$ can sometimes be computed analytically, but often, due to complexity, Monte Carlo methods are needed [BAL12]. The following subsections lay out methods and definitions that are used in this thesis in combination with uncertainty propagation.

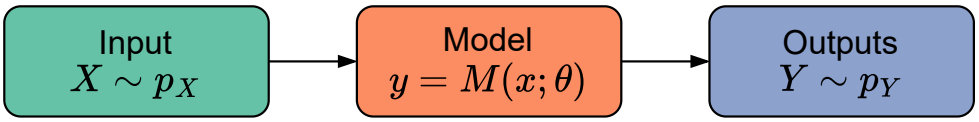


Figure 2.4: Principle of uncertainty propagation adapted from Zhang et al. [Zha21]. A random variable X is distributed according to the probability density function $p_X(x)$. Another random variable at the output Y is defined by the transformation of the model as $Y = \mathcal{M}(X)$.

Monte Carlo Methods

Gillmann et al. [GSWS21] name Monte Carlo sampling as one method to propagate uncertainty. The idea is to estimate a parameter of a random variable by drawing n independent and identically distributed (i.i.d.) samples from its distribution. Zhang et al. [Zha21] define it mathematically as the following. The expected value of the random variable Y is defined as

$$s = \mathbb{E}[Y]. \tag{2.3}$$

s is estimated by $\hat{s}_n$, which is the average of n random samples from the distribution of Y:

$$\hat{s}_n = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.4}$$

The estimator for the mean is unbiased, which means that $\hat{s}_n$ correctly estimates s:

$$\mathbb{E}[\hat{s}_n] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[y_i] = s. \tag{2.5}$$

Furthermore, because of *strong law of large numbers*, the estimator converges to the real expected value:

$$\mathbb{P}\left(\lim_{n\to\infty} |\hat{s}_n - s| = 0\right) = 1 \tag{2.6}$$

And,

$$\mathbb{V}[\hat{s}_n] = \mathbb{E}\left[(s - \hat{s}_n)^2\right] = \frac{\sigma^2}{n} \tag{2.7}$$

shows that the variance σ^2 of the estimator decreases with increasing n. Most importantly is the following equation,

$$\zeta^2 = \frac{1}{n-1}\sum_{i=1}^{n} (y_i - \hat{s}_n)^2 \tag{2.8}$$

that shows how the error ζ^2 of the n samples of a Monte Carlo experiment can be unbiasedly estimated [Zha21], and thus represents a measure of the uncertainty.

Central Limit Theorem

Suppose we have a sequence of i.i.d. random variables $X_1, ..., X_n$, with $\mathbb{E}[X_i] = \mu$ and $\mathrm{Var}(X_i) = \sigma^2 < \infty$, and we are interested in the sample mean $\bar{X}$. Then the Central Limit Theorem (CLT) states that with n going to infinity, the distribution of the random variables $\sqrt{n}(\bar{X}_n - \mu)$ converges to a distribution of $\mathcal{N}(0, \sigma^2)$ [Dur19].

Imagine the setup for the Monte Carlo method again. We can now use CLT to approximate the resulting distribution of the sample mean $\bar{X}$, if the input random variable X distribution is known and the n is sufficiently large. This is beneficial because we calculate confidence intervals of the resulting normal distribution and thus get a better understanding of the data $\bar{X}_n \approx \mathcal{N}(\mu, \frac{\sigma^2}{n})$. A demonstration of the principle of CLT can be seen in Figure 2.5.

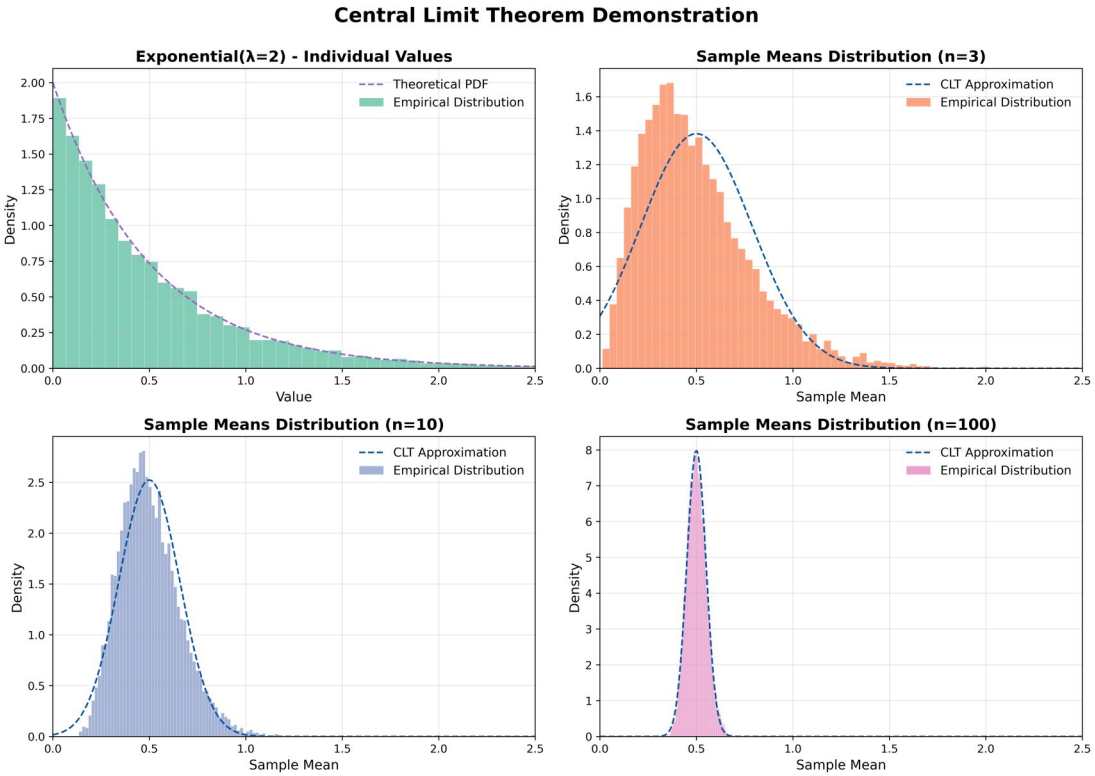


Figure 2.5: Principle of CLT. With increasing n, the distributions of the samples converge to the approximated distribution.

Gaussian Processes

A Gaussian Process (GP) defines a *distribution over functions* [Ras04], and serves as a prior that leverages Bayesian inference to learn from data. Before seeing any data, the GP prior is specified by a mean function (encoding general expectations) and a covariance function or kernel (encoding smoothness, variability, and other structural assumptions). Conditioning this prior on observed data yields the posterior process, which is the trained GP. Figure 2.6 shows a short illustration of this. Despite their flexibility in application, Gaussian Processes (GPs) are computationally infeasible due to the quadratic memory and cubic time complexity. To overcome this, many approximation approaches for GPs have been proposed. In particular, Katzfuss et al. [KG21] present their Vecchia Approximation Framework that is based on the approach by Vecchia [Vec88], which makes approximations of GPs computationally feasible.

Worth mentioning in the context of this thesis is the work of Peischl et al. [PR25], who developed an interactive framework to explore the propagation of CT acquisition artifacts' uncertainty. Their tool allows a user to inject synthetic acquisition artifacts (salt and pepper noise, ring artifact, motion artifact, etc.) into a visualization pipeline

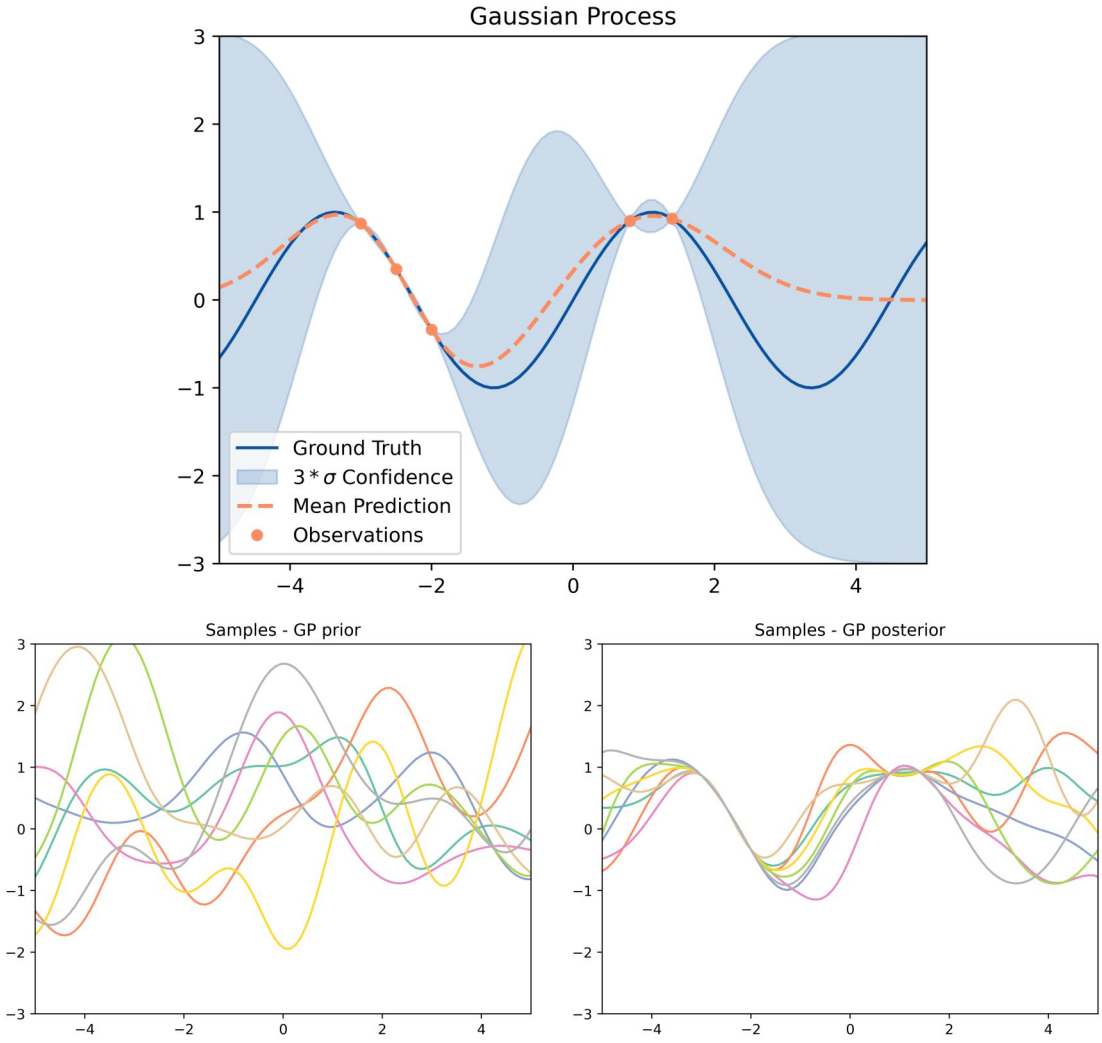


Figure 2.6: The top graph shows the model of a GP that aims to predict a sine. It shows how obtained observations change the confidence intervals of a GP. The lower two graphs show samples before (prior) and after (posterior) "learning" from the observations.

(acquisition and segmentation). In their tool, a user can assign value ranges for all artifact parameters and thus can specify artifact combinations that shall be processed in the pipeline. After all pipeline combination settings are processed, radiomics features are computed for each result and analyzed with principal component analysis and a t-distributed stochastic neighbor embedding. The authors conclude that artifact type, order, and magnitude do impact the resulting segmentation. The work of Peischl et al. [PR25] aims to understand uncertainty propagation of acquisition artifacts. But, their approach shares rather similarities with a parameter sensitivity analysis (see Section 2.3.3), as the approach is based on tweaking input parameters within fixed ranges and analyzing the output. This is inherently different from an uncertainty propagation approach, which would generally try to quantify the distribution of the uncertainty at the output.

2.3 Uncertainty Visualization and Parameter Sensitivity Analysis

2.3.1 Uncertainty Visualization

In order to convey uncertainty information effectively, proper uncertainty visualizations are needed. Therefore, an adequate choice of the uncertainty visualization method for the specific type of data is required. However, uncertainty visualizations are always at risk of introducing too much complexity compared to the same visualization without uncertainty [BHJ+14]. Furthermore, uncertainty often comes to dominate the display, and is sometimes emphasized more than the underlying data [BAL12].

In their survey, Kamal et al. [KDJ+21] divided the state-of-the-art approaches into 6 categories: *Geometry*, *Attributes*, *Animation*, *Visual Variables*, *Graphical Techniques*, and *Glyphs*. Each one will be explained briefly in the following.

Geometry

These approaches visualize uncertainty through the modification of geometry. One approach is to modify existing geometry, like distortion, scaling, or rotation. Another one is to add geometry, such as contours (contour lines, isosurfaces) and volumes.

Attributes

Such methods map uncertainty by changing scene-level rendering parameters, e.g., the surface's shading model (diffuse vs. specular), light intensities or positions, material color, etc.

Animation

Another way to express uncertainty is through animations. Uncertainty can be mapped to various attributes like motion or speed. A benefit of animations is that they tend to reduce visual clutter, since they add a dimension to the visualization. On the other hand, a drawback is that a user might feel overwhelmed. Figure 2.9 shows such uncertainty

animation frames from the work of Lundström et al. [LLPY07], which will be explained further in section 2.3.2.

Visual Variables

Methods in this category encode uncertainty by mapping it to parameters like color, hue, brightness, and saturation are common visual variables to convey uncertainty. Visual semiotics is another type of visual variable. Figure 2.7 shows a series of semiotics used in the study from MacEachren et al. [MRO+12] in which they assess intuitiveness and performance of semiotics in an empirical way.

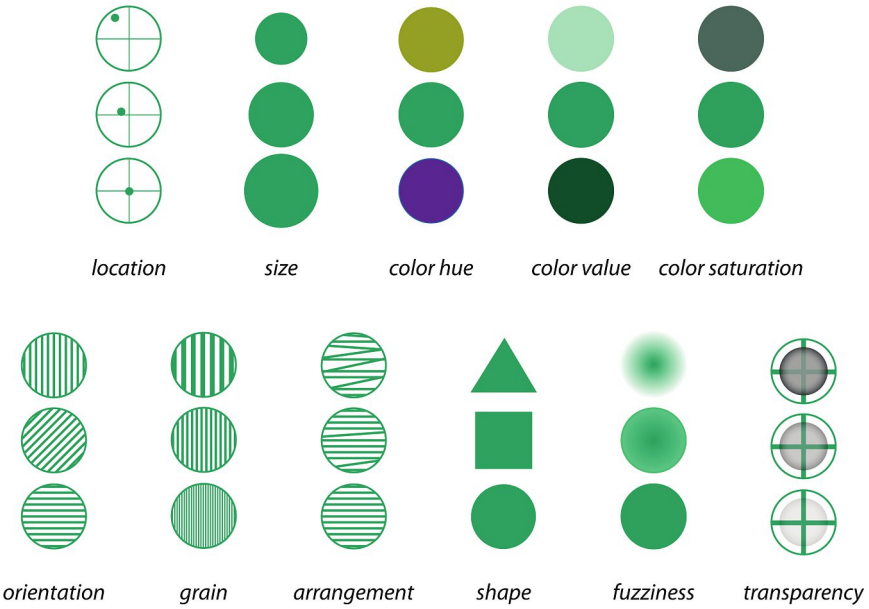


Figure 2.7: Semiotics from MacEachren et al. [MRO+12] to encode uncertainty in different aspects.

Graphical Techniques

Prominent techniques like box plots, scatter plots, or histograms are graphical techniques that offer numerous possibilities to convey data and its uncertainty.

Glyphs

Glyphs are symbols that encode information (e.g., uncertainty) in different properties like shape, size, color, direction, etc. One common base shape is the arrow shape. Since glyphs can encode multiple different characteristics, they are multivariate and therefore well suited to encode uncertainty.

Ouermi et al. [OLM+24] present a three-dimensional glyph called "squid glyph," that encodes directional, rotational, and magnitude uncertainty. In their example Figure 2.8, they use different types of glyphs to show hurricane and wildfire winds and the respective

uncertainty. Their comparison concludes that their novel squid glyph performed best overall, especially in encoding directional variations.

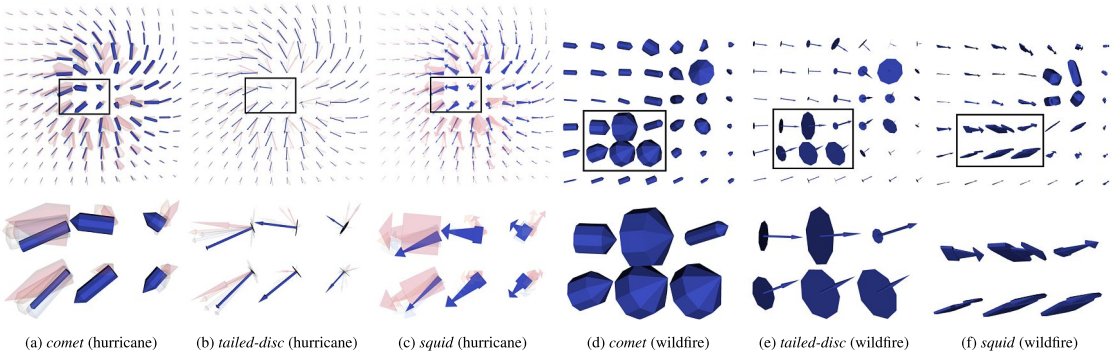


Figure 2.8: Glyph comparison visualization from Ouermi et al. [OLM+24] between comet, tailed-disc and squid glyph. The glyphs encode directional, rotational, and magnitude uncertainty of hurricane and wildfire data.

2.3.2 Uncertainty Visualization in Medical Application

Lundström et al. [LLPY07] propose animation methods to visualize uncertainty in medical volume rendering. Animation frames are created by sampling from a probabilistic transfer function. As a consequence, regions of high uncertainty vary more than certain regions. Furthermore, they present a tool called *sensitivity lens* that can be used in rendering to animate the uncertainty in specified regions. Figure 2.9 shows a usage of the sensitivity lens in angiography. Despite a suspected stenosis in the top left traditional rendering, the uncertainty animation showed in one frame that there is no stenosis. If a vessel is less visible and therefore only perceivable in one of the animation frames, that only means that the contrast agent is less contracted in that region, but still present and thus non-blocked.

Saklani et al. [SGB+24] propose uncertainty-aware implicit neural representations to model scalar fields and study the use of uncertainty estimates for volume visualization tasks. Therefore, they evaluate the effectiveness of two deep uncertainty estimation techniques (Deep Ensemble and Monte Carlo Dropout). They show that uncertainty-aware models provide informative uncertainty visualizations, while providing uncertainty information as a measure of trustworthiness.

Viviers et al. [VVWS23] present a 3D probabilistic segmentation framework that is enhanced with normalizing flows. They train their model on annotated lung lesion datasets that are slightly varying due to different expert annotations. Furthermore, the model then segments lung lesions and estimates the respective uncertainty. Figure 2.10 shows the model segmentation predictions as an uncertainty visualization over multiple slices.

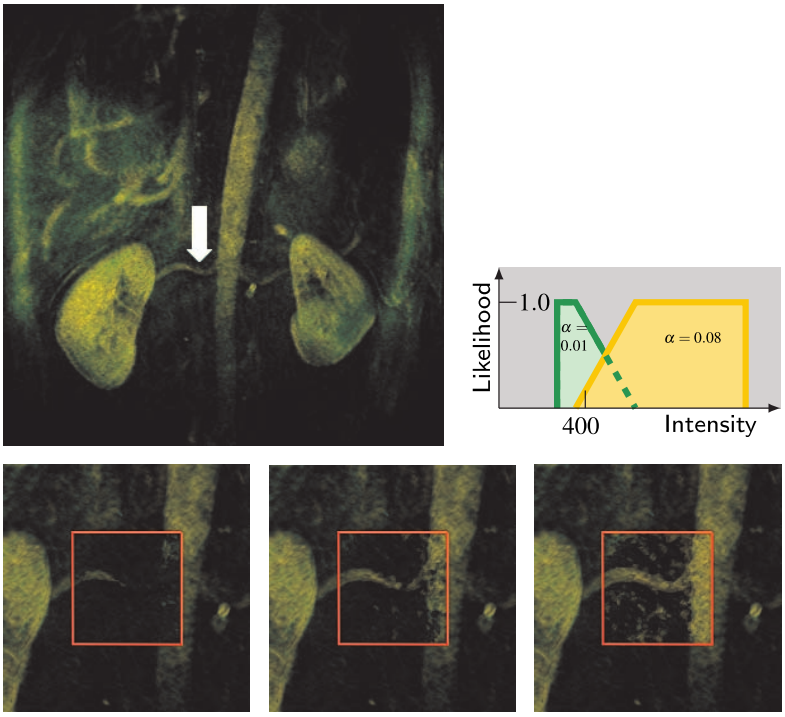


Figure 2.9: Uncertainty animation from Lundström et al. [LLPY07]. The upper left traditional rendering shows a suspected stenosis, while the uncertainty animation frames in the lower row show, despite expectation, a healthy vessel.

2.3.3 Parameter Sensitivity Analysis

In order to understand the term parameter sensitivity analysis, one needs to understand the taxonomy that distinguishes between local and global sensitivity analysis methods [SAB+19]. Local methods evaluate how small input changes influence the model's output, whereas global methods vary uncertainty factors in their entire feasible space to determine the effect of each parameter, in addition to interactive effects (e.g. $x_1 * x_2$) [RHM+22]. Observe that the principle of local methods has similarities to uncertainty propagation methods like (e.g., Monte Carlo), which is pointed out by [FEM+22]. Nevertheless, it is important to understand those concepts and differences of parameter sensitivity analysis (and its two sub-types, local and global analysis), and uncertainty analysis (uncertainty propagation), since they are often mixed and misused [FEM+22, SAB+19]. Another definition from Saltelli et al. [Sal02] defines parameter sensitivity analysis as: "the study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input", which defines the term with more respect to the global side of the analysis. One prominent method for parameter sensitivity analysis is one-at-a-time analysis (local method). For that, each model input changes one at a time, while the other parameters are fixed [BP16]. One-at-a-time sensitivity analysis is a simple and sufficient method to increase the understanding of the model parameters

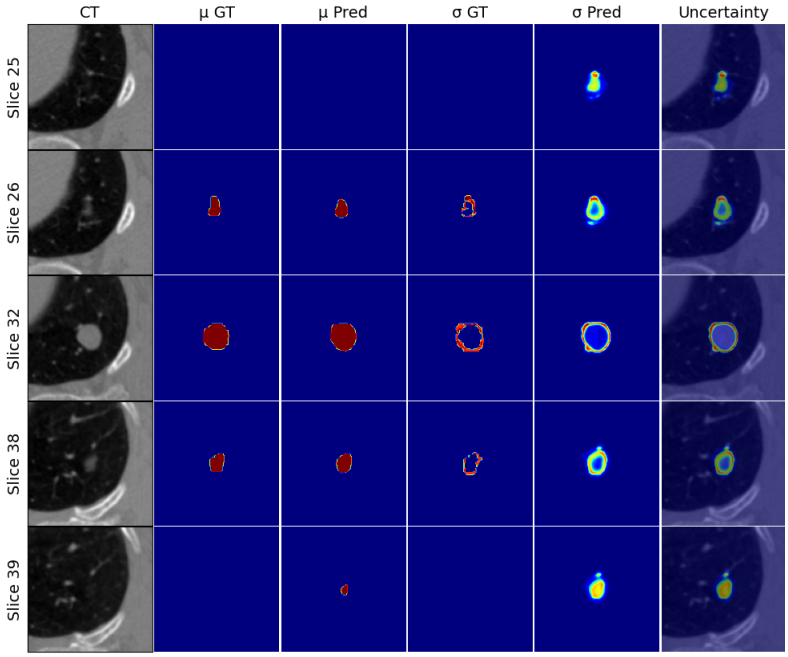


Figure 2.10: Visualization from Viviers et al. [VVWS23]. Each row shows another slice of a lung dataset. The first column shows the CT image, the second and third columns show the mean of the ground truth and the prediction, the fourth and fifth columns show the standard deviation of the ground truth and the prediction, and the last column shows a fusion of the standard deviation of the prediction (uncertainty) and the CT image.

if the inputs of the model are independent and have a linear impact on the output. Because in such a case, one can calculate the influence of each independent parameter with one isolated change. However, models are often more complex and require global methods that exploit the whole parameter space effectively. In their review, Saltelli et al. [SAB+19] investigated models and their respective sensitivity analysis from 280 papers. 7% of the models were linear, 61% nonlinear, and 32% unclear. They conclude that up to 65% of the reviewed papers use inadequate methods. If one can not be completely certain that a model is linear and has no interactions between inputs, global methods should be chosen [RHM+22]. In addition, an uncertainty analysis should be performed before a sensitivity analysis if uncertainties are out of discussion [SAB+19].

2.4 Provenance

2.4.1 Definitions

The term provenance, in general, is defined as "the history of ownership of a valued object or work of art or literature" [Mer25], or as "the place of origin of something" [Cam25]. The World Wide Web Consortium (W3C) defines the *PROV* standard [Wor13c], which

provides a model and definitions that enable interoperable interchange of provenance data across domains. They define provenance as "a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing [Wor13a]." Translated to provenance of visualizations, it is the record of all building steps and different paths that were performed on the visualization in some way. To clarify, it is also about actions that were undone. The following list shows how the building blocks of the *PROV* standard translate to an example of an interactive visualization pipeline:

- Record → Provenance log
- Agents → Interacting users or automated scripts
- Entities → Pipeline states
- Activities → Actions or changes of the pipeline

2.4.2 Provenance Visualization

Provenance can be visualized in various ways. Xu et al. [XOW+20] describe four common encoding schemes for provenance data. The simplest is a *sequence*, an ordered list of the captured interactions. A *grammar*-based encoding infers rules or a domain-specific language from user actions, which aims to automate and reapply processes. A *model*-based encoding, instead of storing raw actions or rules, fits a quantitative model to the user's interactions and analytic state. The final encoding is the *graph*-based approach, where each node represents a state and edges an action taken, which is used for provenance in this thesis for reasons discussed later in the Chapter.

2.4.3 Reasons for Provenance

There are multiple reasons for tracking provenance. Xu et al. [XOW+20] identify six primary "WHY" motivations for provenance data in visualization—and note that fuzzy boundaries exist in this categorization schema, as these categories often overlap and blur. What follows is a discussion of each category, along with some representative examples.

Understanding of the User

The goal is to analyze users' decision-making and create models that describe human reasoning. For example, Musleh et al. [MCER25] measure user confidence against user trust by recording user interactions and assessing the impact of guidance through analysis of the provenance graph using a set of social network metrics. We categorize this work as *understanding of the user*, because they build and evaluate a model for user confidence using provenance data to compute social network analysis metrics.

Evaluation of System and Algorithms

The goal is to analyze users' actions to gain insights into a system's or algorithm's performance and thus enable evaluation. For example, Blaschek et al. [BJK+16] create a visual analytics system that aims to evaluate interactive visualization systems, with provenance data recorded from interaction, thinking aloud, and eye movement. Thus, analysts can search for patterns in the combined data and verify hypotheses. We classify their work as *evaluation of system and algorithms* because their visual analytics approach is designed to evaluate interactive visualizations.

Adaptive Systems

The objective is to use provenance to recommend visualizations, guide their construction, or prefetch data for likely next visualizations. For example, Battle et al. [BCS16] present ForeCache, a tool to explore and browse large datasets. To improve response times, ForeCache prefetches data the user might need in the future by performing predictions based on the user's previous interactions and data characteristics. We classify this *adaptive systems*, because the system uses provenance data of the visualization itself to improve the system's performance.

Model Steering

The aim is to improve underlying models (e.g., machine learning) based on users' interactions with the visualization. For example, Endert et al. [EFN12] present their prototype ForceSPIRE, which learns the clustering of text documents from the captured interaction provenance. The tool updates the underlying model as the user performs cluster changes and performs the respective layout changes accordingly. We classify this as *model steering* because an underlying model is trained on the provenance data of the visualization.

Replication, Verification, and Re-Application

The goal is to record provenance so analyses can be replayed or applied to new datasets. This allows verification or replication of results. For example, Stitz et al. [SLSG16] present with Adaptive Visualization of Comprehensive Analytical Data Origins (AVOCADO) a provenance visualization approach that aims to ease exploration and reproducibility in biomedical research. They do so by aggregating provenance data based on an adaptive degree of interest heuristic, which is furthermore inferred from the interaction of the user with the provenance graph. We classify the tool as *replication, verification, and re-application*, because its motivation is to support reproducible research, which they simplify with the abilities of AVOCADO. Figure 2.11 shows the overview of AVOCADO.

Report Generation and Storytelling

The goal is to tell the story of how a visualization was created. Therefore, the focus is to reduce provenance data into meaningful chunks and generate reports to summarize the relevant parts. For instance, Schreiber et al. [SS17] present a visualization technique that aims to convey the information of provenance data with the help of comics. To prove their concept, they developed a prototype that automatically creates comic strips out of PROV-compliant documents. We classify this as a prime example of *report generation and storytelling* as they transform complex provenance information into simple visuals.

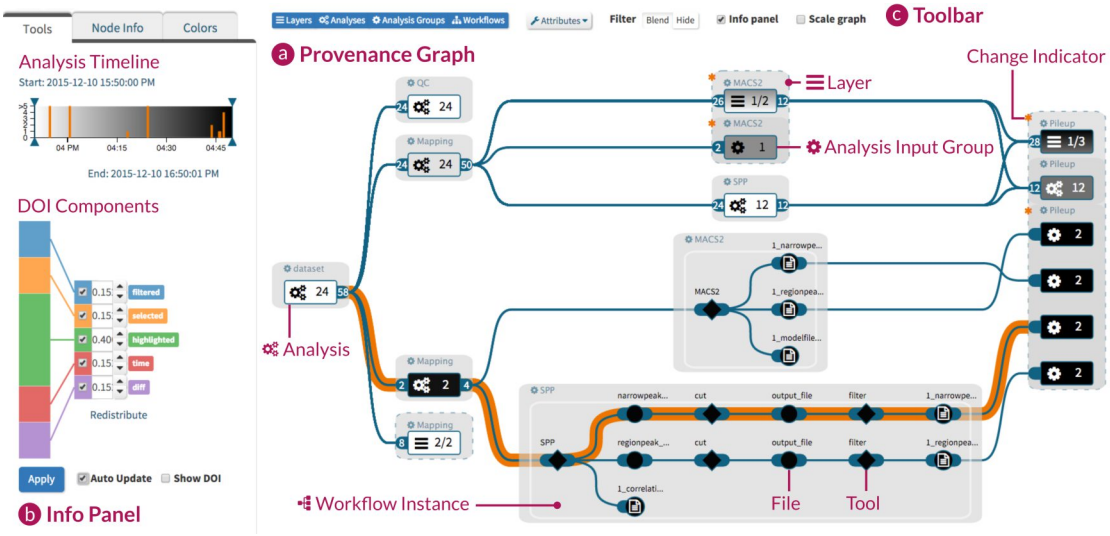


Figure 2.11: Visualization of AVOCADO from Stitz et al. [SLSG16]. Part (a) shows the aggregated and filtered provenance graph. Part (b) defines how the degree-of-interest heuristic function is weighted, which is used for aggregation.

Figure 2.12: A graphical representation of provenance data from Schreiber et al. [SS17]. The graph is designed according to the PROV-PRIMER [Wor13b] standard. The data describes two actions: first, downloading step data from a cloud service, and second, visualizing the data.

The corresponding provenance data and the comic strip can be seen in Figure 2.12 and Figure 2.13.

Another example for *report generation and storytelling* is the work of Amabili et al. [AKM+18], where they record provenance data of radiology workflows and ease exploration as they provide an interactive visualization together with a provenance graph, where it is possible to group by different levels of abstraction. The aim is to adjust the workflow's level of detail based on the expertise of the current user. Examples of such grouped provenance graphs and their storytelling tool can be seen in Figure 2.14.

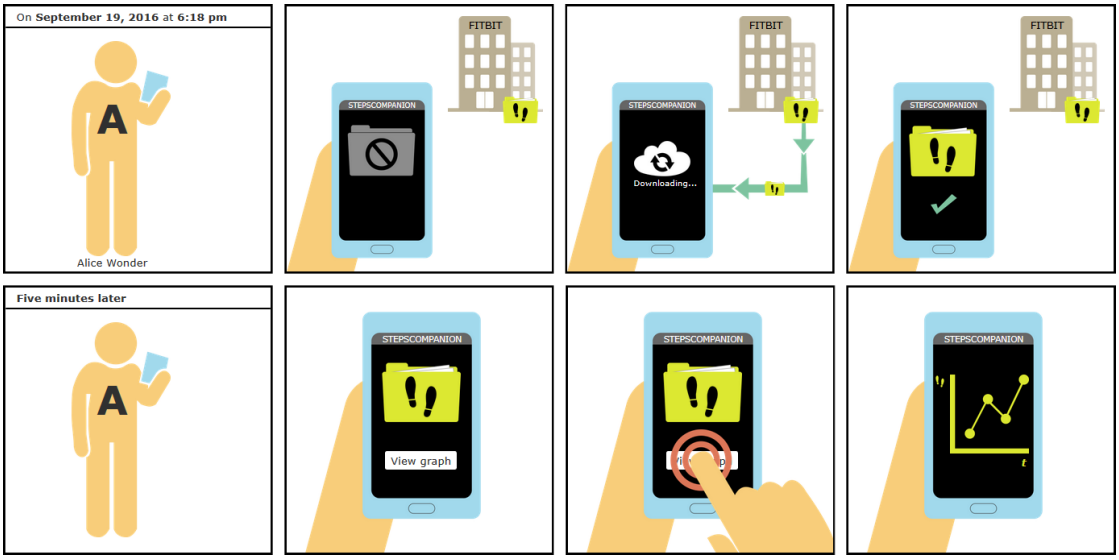


Figure 2.13: The comic strip provided by Schreiber et al. [SS17], generated with the provenance data in Figure 2.12. The comic depicts the actions of the provenance data as much more accessible.

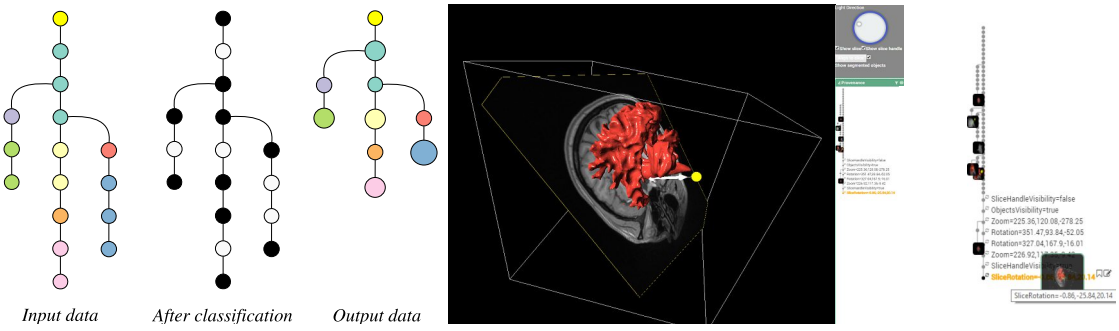


Figure 2.14: Figure provided by Amabili [AKM+18]. *Left figure*: shows how provenance data can be grouped in a graph. Similar color means the same type of action. Then, in the middle, black nodes are key nodes, whereas white ones are regular nodes that can be collapsed. This results in a grouped graph on the right. *Right figure*: Shows on the left side the storytelling tool with the visualized medical data and on the right side the underlying provenance workflow.

2.5 Summary of the State of the Art and Identified Gap

All in all, we have seen that a lot of research effort has been put into uncertainty classifications [GSo06, THM+05, PWL97b, SLSR10]. Furthermore, potential uncertainty sources have been identified and classified [GSWS21, RPHL14]. Many uncertainty propagation methods exist that try to estimate the propagated uncertainty [GSWS21]. Moreover, provenance solutions are used in many different research departments for various reasons [XOW+20]. Despite advances in research, the state of the art in uncertainty quantification and visualization lacks standardized methodologies that enable users to assess and understand the accumulation and propagation of uncertainties across multi-step processes, such as the visualization pipeline [PR25]. Additionally, there is a need for provenance solutions that effectively track the propagation and accumulation of this uncertainty throughout the visualization pipeline. This thesis aims to address these two gaps in the literature by proposing an interactive visual analysis framework that supports the comparison of uncertainty propagation results from distinct provenance pipeline states.

CHAPTER 3

Methodology

3.1 General Idea

In this section, we introduce our framework using Figure 3.1, which comprises two core components, the data pipeline *Part A*, and the provenance tree *Part B*, and shows in *Part C* how these components interact during execution.

A user shall be able to use the framework to build a modular visualization pipeline and add uncertainties wherever they are suspected (see *Part A*). Thereby, the options are either a Parameter Uncertainty (PU) (see the yellow circle in *Part A*) or Visual Uncertainty (VU) (see the orange circle in *Part A*). Subsequently, every change to the pipeline is captured and visualized in the provenance tree (see the green rectangle in *Part C*) as pipeline states. A user can use those states to rapidly jump back and forth between them and reload specific pipelines if wanted (see the pink rectangle in *Part C*).

If a pipeline includes uncertainties, it is run n times, whereby in each run, each uncertainty is resampled. By calculating statistics (mean and variance) over all samples, the framework computes the node impact, which is stored in the respective pipeline state's node (see the orange rectangle in *Part C*). Simply put, a node impact computes for each uncertainty present in the pipeline state how the results of the pipeline would have looked if the uncertainty had been left out. Thus, a user can compute the node impact for all nodes of interest and compare them to get insights into how the pipeline changes affected the results (see the blue rectangle in *Part C*).

The following sections provide a detailed description of the concepts behind the development of the framework. The components are depicted in Figure 3.1 and will be discussed in the following sections.

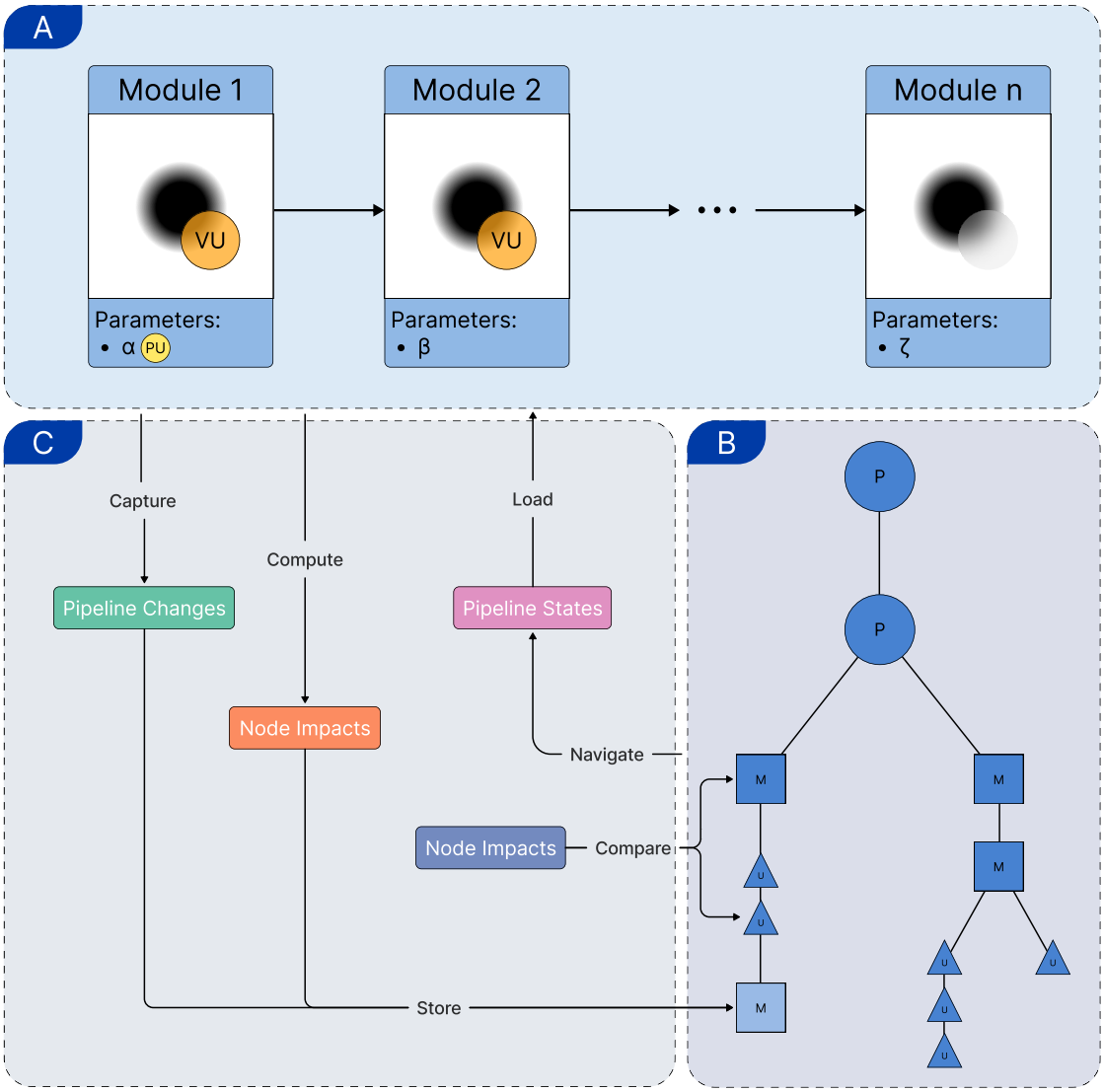


Figure 3.1: Conceptual overview of the framework. *Part A* represents the current modular pipeline with its Parameter Uncertainties (PUs) (yellow) and Visual Uncertainties (VUs) (orange). *Part B* shows the provenance graph, which is a hierarchical tree. *Part C* shows the most important functionality and interaction between the pipeline and the provenance tree. The provenance tree captures changes from the pipeline and stores them as nodes in the graph. If the node impact for the current pipeline is computed, it is stored in the respective node. Such node impacts can be selected in the provenance graph and compared to gain insights. Moreover, the provenance tree can be used to control the currently loaded pipeline state by navigating in the provenance tree.

3.2 Pipeline

This paragraph discusses the pipeline part of the developed framework, which corresponds to *Part A* in Figure 3.1. To provide the user with a flexible way to build a pipeline, it is helpful to offer high modularity in the pipeline-building process. Therefore, each pipeline created with the framework consists of exactly one acquisition module at the beginning and an arbitrary number of transformation modules. Classic pipelines have visualization modules at the end. However, in our case, we include a fixed visualization module for each module. If the user wants to simulate uncertainty in the pipeline, uncertainties can be introduced at any module in the pipeline and can additionally be inspected.

3.2.1 Modules

Modules are the building blocks for the pipeline. Due to modeling reasons, we refer to the actual operational functionality of the module as the *processing unit*. Since the purpose of the framework is to function as a proof of concept, basic imaging operations for the processing units were chosen. These could be further expanded in the future by including more complex operations. We divide them into the stages of acquisition, transformation, and visualization, depicted previously in Figure 2.2. More specifically, we model a module as follows: Each module has an input and output image, a view element that visualizes the output image, a collection of PUs and VUs, and a processing unit. The following describes the available processing units.

Acquisition

Circle Generator (s, r) : integer s is the side length of the image; integer r is the radius of the circle

The *Circle Generator* is for simple exploration or testing purposes. It generates a grayscale image with a resolution of $s \times s$ that contains a centered circle with a radius r and intensity of 255 on a 0 intensity background.

Image Acquisition (p) : image dataset location path p

The *Image Acquisition* processing unit offers high flexibility due to the ability to load 2D images from a specific path p. For medical imaging, it also allows users to explore datasets by scrolling through the images and offers the option to deactivate the scaling of DICOM images.

Transformation

Gaussian Filter (σ) : positive real number σ is the standard deviation for the gaussian kernel

The *Gaussian Filter* acts as a low-pass filter. The convolution of an image with a Gaussian kernel is a weighted sum depending on σ of neighboring pixels, and thus dampens (blurs) high-frequency parts of the image. Higher σ values yield stronger blurring.

Median Filter (w) : integer window size w

The *Median Filter*, unlike the Gaussian filter, is a nonlinear filter. For each pixel, you select a $w \times w$ neighborhood around it, sort those values, and replace the center pixel with the median. This introduces another kind of blur, which is often used to remove noise.

Laplace Filter ():

The *Laplace Filter* is often used for edge detection in image processing. It is defined as the sum of second derivatives of often a 3×3 window.

Morphology Filter (w, t): integer w is the window size; t morphology operation type, either *erosion* or *dilation*

Morphology operations work analogously to the *Median Filter*. Instead of selecting the median, the minimum value of the window is picked in the case of *Erosion* and the maximum value in the case of *Dilation*. Morphology operations are often applied in cascade, i.e., *Opening* is equivalent to *Erosion* $\rightarrow$ *Dilation* and *Closing* is equivalent to*Dilation* $\rightarrow$ *Erosion*.

Threshold Segmentation (l, u): integer l is the lower threshold bound; integer u is the upper threshold bound

Threshold Segmentation is the simplest form of thresholding, yet it can be highly effective. The upper and lower thresholds define an intensity interval, and any pixel whose intensity falls into that interval is selected.

Otsu Threshold Segmentation ():

The *Otsu Threshold Segmentation* method from Otsu [Ots79] computes a threshold k that splits the intensity histogram into two classes so that the weighted intra-class variance is minimized.

Visualization

Simple Render ():

The *Simple Render* processing unit functions as a placeholder for further extensions, as it only displays the 2D data.

Each of the explained processing units is illustrated in Figure 3.2 in an example.

3.2.2 Uncertainty

Uncertainty Model

To simulate uncertainty propagation, the framework requires a mathematical model that captures the stochastic nature of input parameters and therefore needs to make assumptions on the uncertainty distributions [Zha21]. Our uncertainty model is defined by

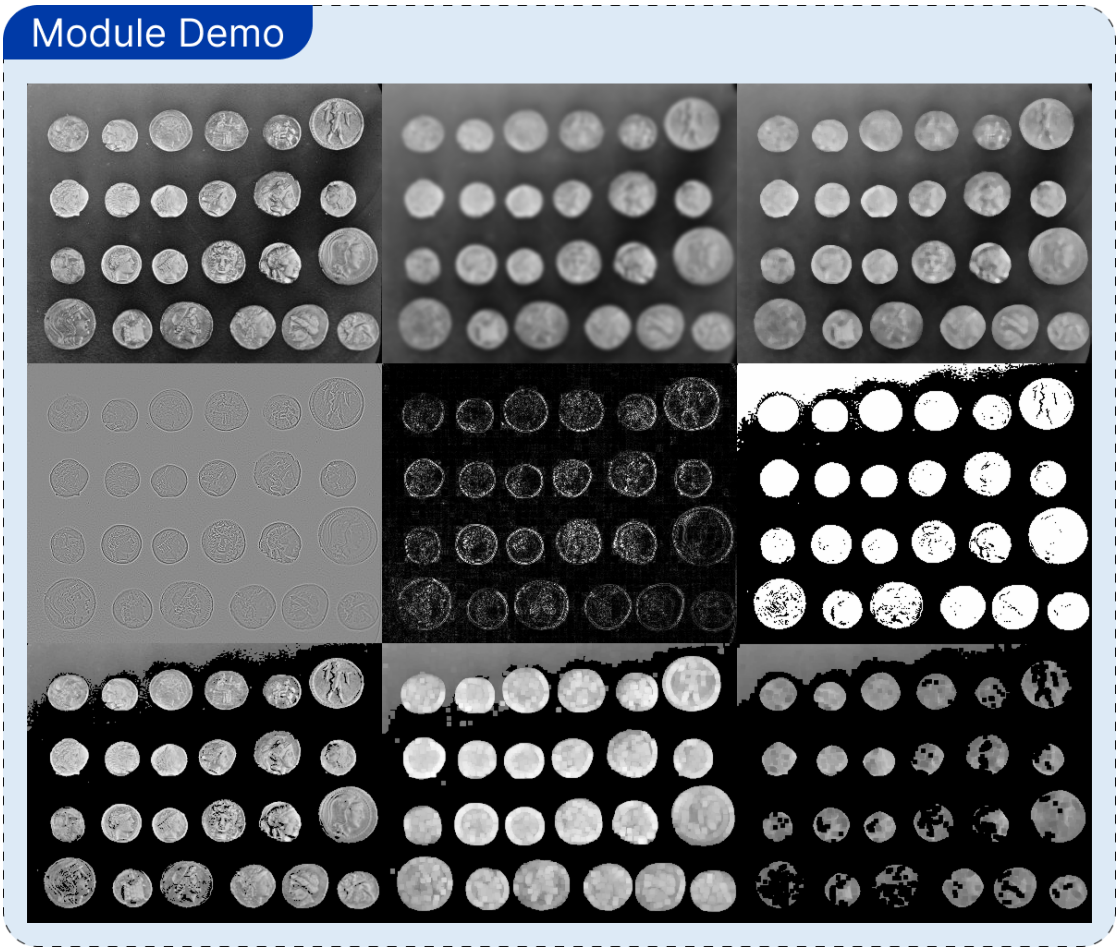


Figure 3.2: Demonstration of all ■ Transformation stage processing units that are available in the framework. The example image used is from scikit-image by Walt et al. [WSNI+14]. *Top left:* Original image, *Top center: Gaussian Filter* with $\sigma = 2$, *Top right: Median Filter* with $w = 5$; *Center left: Laplace Filter* with a individual Lookup Table (LUT) setting, *Center center:* Same *Laplace Filter* but with the same LUT as the other images, *Right center: Otsu Threshold Segmentation*; *Bottom Left: Threshold Segmentation* with $l = 105$ and $u = 252$, *Bottom center: Morphology Filter* Dilation operation with $w = 5$, *Bottom right: Morphology Filter* Erosion operation with $w = 5$.

four key parameters: the mean-offset μ_0, the mean-spread s_σ, the noise-spread s_n, and the distribution type t. These parameters collectively define the uncertainty characteristics for each variable in the simulation.

1. **Distribution type** t: Specifies the probability distribution assumed for an uncertainty. The model supports three distribution types: normal, uniform, and beta distribution.
2. **Mean-offset** μ_0: defines the offset from the mean of the data and can be either absolute or relative.
3. **Mean-spread** s_μ: defines the absolute or relative spread of the mean and is primarily relevant for VU, which are explained later on.
4. **Noise-spread** s_n: specifies the relative or absolute spread of the uncertainty.

The concept of the model is that, independent of the distribution type, the uncertainty values shall be in the interval:

$$a = \mu + \mu_0 - s_\mu - s_n, \tag{3.1}$$
$$b = \mu + \mu_0 + s_\mu + s_n, \tag{3.2}$$
$$I = [a, b]. \tag{3.3}$$

Nevertheless, it is not possible to express all distributions into that interval because not all distributions have finite support (e.g., normal distribution). Therefore, we define the three distributions as follows.

Independent of the mean-spread or noise-spread, we define a spread s as the base deviation

$$\sigma_{base} = s/3. \tag{3.4}$$

For the normal distribution, we can apply the $3 * \sigma$ rule with

$$\mathcal{N}(\mu, \sigma_{base}^2), \tag{3.5}$$

which means that $\sim 99.7\%$ of the samples are in the interval I. The uniform distribution is constructed by using the bounds of I:

$$\mathcal{U}(a, b) \tag{3.6}$$

The beta distribution is defined by two parameters α and β, which, together, define the shape of the distribution. In the framework, those parameters are fixed to $\alpha =$

0.1, and $\beta = 0.1$ and then scaled to I by multiplying with the interval length and adding the minimum:

$$X = a + (b - a) \cdot Y, \text{ where } Y \sim \text{Beta}(\alpha, \beta) \tag{3.7}$$

We chose the beta distribution with those parameters to get a different option from the normal and uniform distributions.

If multidimensional data is sampled, the mean-spread value s_μ defines how the mean of the sample is spread, while the noise-spread defines how each single data point varies. Note that the distribution type for both spreads is the selected type t. In the case of one-dimensional data, there is no real benefit of having two different spread settings other than being able to define two distributions of the same type and convolve them (see Figure 3.3). Algorithm 3.1 is for clarification of the principle of the mean-offset μ_0 and the spread s, which both apply for the mean-spread and noise-spread. Depending on each setting, the values are interpreted as absolute or as a fraction of the reference value p_{ref}.

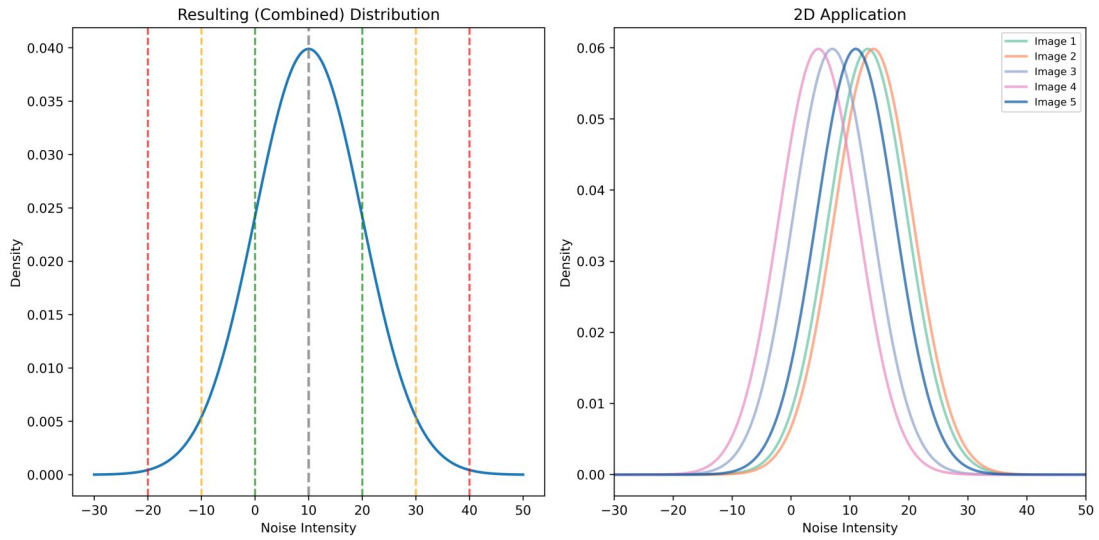


Figure 3.3: Distribution demonstration for an uncertainty model with the parameters: $\mu_0 = 10$, $s_\mu = 10$, $s_n = 20$, $t = uniform$. On the left side is the resulting distribution where 99.7% of the values are between -10 and 30. This distribution is applied in case of a PU. On the right side is the corresponding distribution of a 2D or VU case. Each curve in the plot represents one sample for a region. Each region (each curve) receives an offset between -10 and 10 due to s_μ and additionally has a noise distribution that spans from -20 to 20.

Algorithm 3.1: Uncertainty Model

Input: Mean-offset μ_0, spread s, absolute flags μ_{abs}, s_{abs}, reference value p_{ref}
Output: Computed mean-offset μ_{0c}, computed spread s_c

if $\mu_{abs} = true$ **then**
  $\mu_{0c} \leftarrow \mu_0$;
else
  $\mu_{0c} \leftarrow p_{ref} \times \mu_0$;
end
if $s_{abs} = true$ **then**
  $s_c \leftarrow s$;
else
  $s_c \leftarrow p_{ref} \times s$;
end
return μ_{0c}, s_c;

In literature, Monte Carlo methods are most prominent for uncertainty propagation [GSWS21, Zha21]. Nevertheless, in this work, there have been approaches to use GPs to propagate uncertainty, which would have included to learn how each module affects and propagates the uncertainty measures. Monte Carlo may be able to provide such information, but since the whole framework is a proof of concept, we employed the basic Monte Carlo method on its own (see Section 2.2.4), although there are various more advanced Monte Carlo approaches. Concerning uncertainty, the framework allows placement at any position in the pipeline. Moreover, we categorize uncertainty based on where it is assigned, distinguishing between VU and PU, which both inherit the previously defined uncertainty model.

Visual Uncertainty

Visual uncertainty can be added by placing a shape onto any module's image, which defines uncertainty in the specified region. The magnitude of the uncertainty is controlled by the parameters of the uncertainty model. This can be seen in Figure 3.1 in *Part A* as the orange circles. The shapes for a VU supported by the framework are circles and rectangles. In this way, a user can mark areas of uncertainty by specifying the area of concern and the degree of uncertainty in the parameters of the uncertainty model.

Parameter Uncertainty

A PU can be added to the parameters of modules to simulate uncertainty, which can be seen in Figure 3.1 in *Part A*, representing yellow circles at the parameters. When added, the framework employs an uncertainty model that defines the characteristics of the uncertainty.

Uncertainty Propagation

To apply Monte Carlo methodology and thus propagate uncertainty, we need to run the whole pipeline n times with different sets of parameters. That means that in each pipeline run, elements of uncertainty are replaced with a sample from the respective specified distribution. In case of a VU that sample represents an area of the image that is adjusted, and in case of PU the sample is just another value for the parameter. The following three sections describe the methodology for how the framework propagates uncertainty in more detail.

Step 1: Prepare Pipeline Parameter-Sets

Algorithm 3.2 shows the concept of this step in detail. We start with the current pipeline modules and the sample size. Then we iterate through every module in the current pipeline state to gather its parameters, VUs, and PUs (see lines 5–10), then assemble n complete pipeline parameter-sets in a structure with n rows, each row containing, for each module, the specific parameter list required to process that module's processing unit. So, each row consists of a module parameter set for each module. Plain parameters without uncertainty retain the same value in all rows. For each PU, we generate n samples (see lines 17–22) and place them into the n rows at their respective destination in the data structure (see line 25). VUs require a different approach since they may depend on the mean of the area they belong to in their respective pipeline execution. Therefore, we cannot precompute actual visual samples. Instead, we generate n random seeds (see lines 13–16) for the later process of sampling, ensuring reproducibility and deferring sample generation until the corresponding image data are available.

Step 2: Run Pipelines

For each row in the pipeline parameter-sets, the pipeline is run a single time. Algorithm 3.3 shows the concept of such a run. It does so with each module's processing unit, a row of the pipeline parameter-sets, the shape of the image, and the list of present VUs. The first step is to initialize the output data structure that will hold the output of each processing unit (see line 1). Then we go through all processing units one by one and pass, each time, the respective module parameter set from the current pipeline parameter set to the current processing unit, process, and store the result at the related output (see lines 3–7). If the current module has VUs, we create a sample for the respective area with the corresponding precomputed seed and apply it to the output (see lines 8–21). Note that in case of multiple VUs in one module, the change of all VUs from the module is computed independently from each other and then applied as a whole to the module's output. Otherwise, we would introduce dependencies between VUs, which is not wanted.

Step 3: Evaluation and Statistics

The goal of uncertainty propagation is to get a measure for the uncertainty itself and determine how it evolves along the pipeline. Therefore, we need to calculate the variance of the results in each pipeline run. We do so with the help of the algorithm from Welford [Wel62], which is suited to calculate the mean and variance on the loop by updating the

Algorithm 3.2: Pipeline Parameter-Sets Preparation

Input: Pipeline modules $\mathcal{M} = \{m_1, m_2, \ldots, m_k\}$, sample size n
Output: Pipeline parameter sets $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ where $S_i[j]$ is parameter dictionary for module j in sample i

seed $\leftarrow$ GetRandomSeed();
$\mathcal{S}_{\text{fixed}} \leftarrow \emptyset$;
$\mathcal{S}_{\text{uncertain}} \leftarrow \emptyset$;
$\mathcal{U} \leftarrow \emptyset$ `// Uncertainty objects by module`
for *each module* $m_j \in \mathcal{M}$ **do**
  $\mathcal{U}_{\text{visual}} \leftarrow$ GetVisualUncertainties(m_j);
  $\mathcal{U}_{\text{param}} \leftarrow$ GetParameterUncertainties(m_j);
  $\mathcal{U}[j] \leftarrow \mathcal{U}_{\text{visual}} \cup \mathcal{U}_{\text{param}}$;
  $\theta_{\text{fixed}} \leftarrow$ GetFixedParameters(m_j,$\mathcal{U}_{\text{param}}$);
  $\mathcal{S}_{\text{fixed}} \leftarrow \mathcal{S}_{\text{fixed}} \cup \{(i, \theta_{\text{fixed}})\}$;
  for *each uncertainty* $\omega \in \mathcal{U}[j]$ **do**
    SetSeed(seed);
    if ω *is visual uncertainty* **then**
      $\mathbf{s} \leftarrow$ GenerateRandomIntegers(n);
      $\mathcal{S}_{\text{uncertain}} \leftarrow \mathcal{S}_{\text{uncertain}} \cup \{(j, \omega, \mathbf{s})\}$;
    end
    else if ω *is parameter uncertainty* **then**
      $\theta_{\text{current}} \leftarrow$ GetCurrentValue(m_j, ω);
      $\mu_{0c}, s_{\mu c}, s_{nc} \leftarrow$ GetParameterDistribution($\omega, \theta_{\text{current}}$);
      $\mathbf{s} \leftarrow$ SampleFromDistribution($\mu_{0c}, s_{\mu c}, s_{nc}, \omega, n$);
      $\mathcal{S}_{\text{uncertain}} \leftarrow \mathcal{S}_{\text{uncertain}} \cup \{(j, \omega, \mathbf{s})\}$;
    end
  end
end
$\mathcal{S} \leftarrow$ CreatePipelineSets($n, \mathcal{S}_{\text{fixed}}, \mathcal{S}_{\text{uncertain}}$);
return $\mathcal{S}$;

Algorithm 3.3: Run Pipeline Single Time

Input: Processing units $\mathcal{P}$, S_i row of pipeline parameter sets S, image shape $s = (H, W)$, visual uncertainties $\mathcal{V}$
$\mathcal{P}$: set of $|\mathcal{P}|$ processing units (modules)
$S_i[j]$: module parameter set for module j
$\sigma = (H, W)$: spatial dimensions of output images
$\mathcal{V}$: visual uncertainties of pipeline by their respective identifier
Output: Pipeline run module results $\mathcal{X} \in \mathbb{R}^{|\mathcal{F}| \times H \times W}$

$\mathcal{X} \leftarrow \mathbf{0} \in \mathbb{R}^{|\mathcal{P}| \times H \times W}$;
for $j = 0$ **to** $|\mathcal{P}| - 1$ **do**
  if $j = 0$ **then**
    $\mathcal{X}[j] \leftarrow \mathcal{P}[j](\text{null}, S_i[j])$;
  else
    $\mathcal{X}[j] \leftarrow \mathcal{P}[j](\mathcal{X}[j-1], S_i[j])$;
  end
  if $\mathcal{V} \neq \mathit{null}$ **then**
    $Y \leftarrow \mathbf{0} \in \mathbb{R}^{H \times W}$;
    for $\omega \in \mathcal{V}$ **do**
      if $\mathcal{V}[\omega].\mathit{index} \neq j$ **then**
        continue;
      end
      $s \leftarrow \text{GetSeed}(S_i[j], \omega)$;
      SetSeed(s);
      $\mu_{0c}, s_{\mu c}, s_{nc} \leftarrow \text{GetDistribution}(\mathcal{X}[j], \omega)$;
      $X \leftarrow \text{SampleArea}(\mu_{0c}, s_{\mu c}, s_{nc}, \omega)$;
      $Y \leftarrow Y + X$;
    end
    $\mathcal{X}[j] \leftarrow \mathcal{X}[j] + Y$;
  end
end
return $\mathcal{X}$;

respective values with each sample. The concept of usage in the framework is shown in Algorithm 3.4. While looping over the pipeline parameter-sets (see line 4), we pass each pipeline run result to the algorithm and therefore add its influence to the measures (see lines 5–9). Resulting in an average mean, average variance, and average standard deviation image over all pipeline runs for each module (see lines 11–15).

While the average mean shows us how the module results look on average, the average variance, respectively, shows the variation of each pixel across all modules. In the average mean result, each pixel represents the pixel-wise sum of the averaged samples. This is the premise for CLT, which tells us that with growing n the the average is normal distributed as (see Section 2.2.4):

$$\mathcal{N}(\hat{X}_n, \frac{\hat{\sigma}^2}{n}). \tag{3.8}$$

Therefore, we can compute confidence intervals around each averaged pixel with the error probability α and thus the corresponding quantiles for the interval $\pm z_{1-\alpha/2}$, which are scaled by the standard deviation and placed around the sample mean $\hat{X}_n$.

$$\left[\hat{X}_n - z_{1-\alpha/2}\,\frac{\hat{\sigma}}{\sqrt{n}}\;,\;\hat{X}_n + z_{1-\alpha/2}\,\frac{\hat{\sigma}}{\sqrt{n}}\right]. \tag{3.9}$$

As the framework computes the pixel-wise mean and standard deviation over the course of the sampling phase, we can utilize those statistics to calculate a lower and upper bound confidence interval image for each respective stage on the fly.

Node Impact

To investigate the impact of uncertainties, we need to assess the effect of each uncertainty on the pipeline. For simplicity within this proof-of-concept solution, we choose to employ a one-at-a-time approach, which is illustrated in Figure 3.4. In the example, there are a total of six uncertainties, indicated by the colored circles in the modules. Six uncertainties result in seven runs of the whole pipeline, each of which is run n (sample size) times, depicted in the rows of the figure. In the first six runs, each uncertainty is left out one time, and in the last run, all uncertainties are included for the baseline. Let $\#r$ be the number of pipeline runs the framework has to take to determine the node impact. Furthermore, let $\#v$ and $\#p$ be the number of VUs and PUs present in the pipeline. Then the pipeline must be run once for each present uncertainty that is left out, plus once with all uncertainties to obtain the baseline. This can be mathematically expressed as:

$$\#r = \#p + \#v + 1 \tag{3.10}$$

The Algorithm 3.5 shows the same concept for clarification. The result of that approach is called *Node Impact*. For further analysis, we compute:

Algorithm 3.4: Welford's Online Algorithm for Mean and Variance

Input: Results of Pipeline runs $\mathcal{X} \in \mathbb{R}^{n \times M \times H \times W}$ where n = sample size, M = module count, $H \times W$ = image dimensions

Output: Module-wise mean $\mu \in \mathbb{R}^{M \times H \times W}$, variance $\sigma^2 \in \mathbb{R}^{M \times H \times W}$, standard deviation $\sigma \in \mathbb{R}^{M \times H \times W}$

$n \leftarrow 0$;
$\mu \leftarrow \mathbf{0} \in \mathbb{R}^{M \times H \times W}$;
$S \leftarrow \mathbf{0} \in \mathbb{R}^{M \times H \times W}$;
for *each new sample* $\mathcal{X}_i$ *in* $\mathcal{X}$ **do**
  $n \leftarrow n + 1$;
  $\delta \leftarrow \mathcal{X}_i - \mu$;
  $\mu \leftarrow \mu + \frac{\delta}{n}$;
  $\delta_2 \leftarrow \mathcal{X}_i - \mu$;
  $S \leftarrow S + \delta \cdot \delta_2$;
end
if $n \geq 2$ **then**
  $\sigma^2 \leftarrow \frac{S}{n-1}$ `// Unbiased variance`
  $\sigma \leftarrow \sqrt{\sigma^2}$;
end
return μ, σ^2, σ;

$$\mu_{diff} = R_{full}.\mu - \mathcal{R}[u_k].\mu \;\; and \;\; \sigma^2_{diff} = R_{full}.\sigma^2 - \mathcal{R}[u_k].\sigma^2. \tag{3.11}$$

Here R_{full} is the result of the pipeline run with all uncertainties included. $.\mu$ then is the pixel-wise mean and σ^2 the pixel-wise variance of the corresponding result. $\mathcal{R}[u_k]$ is the result of the pipeline run in which the uncertainty u_k is excluded.

Note that each result R has the dimensions $M \times H \times W$, representing the average result for each module in the pipeline. Furthermore, we use the last layer ($[M-1]$) of μ_{diff} and σ^2_{diff} so we can compute their mean and standard deviation:

$$\mu_{bias} = mean(\mu_{diff}[M-1]) \text{ and } \sigma_{bias} = std(\mu_{diff}[M-1]). \tag{3.12}$$

The μ_{diff} refers to the difference between two complete pipeline runs in each module and thus how a specific uncertainty changed the result of each module. Analogously, the σ^2_{diff} quantifies how the variance of each module changed.

To save computational resources, μ_{bias}, σ_{bias}, μ_{var}, and σ_{var} are only computed for the result of the last module. μ_{bias}, σ_{bias} describe the distribution of the intensities of the bias that came with the addition of the respective uncertainty. μ_{var} and σ_{var} are measures for the mean variance change and therefore denote how the uncertainty evolved on average.

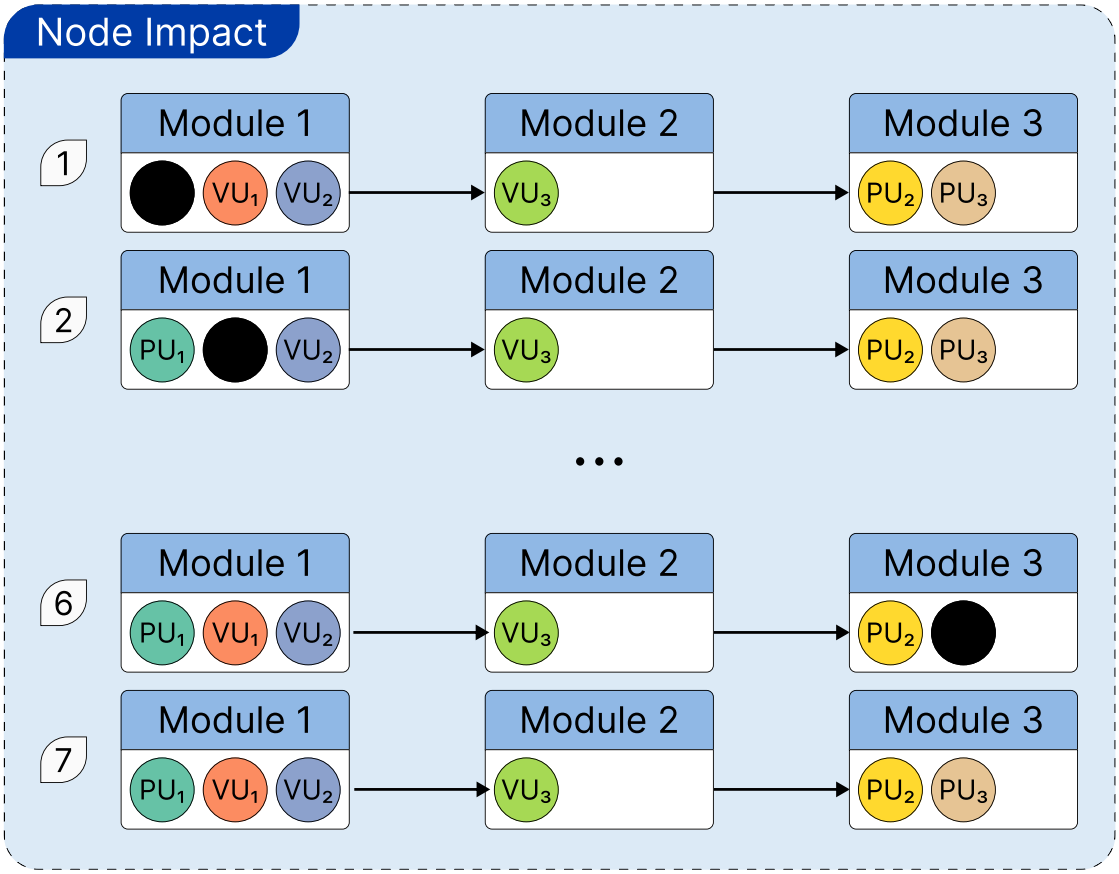


Figure 3.4: Illustration on how a *Node Impact* is computed. In the example, we have six uncertainties distributed across three modules. To compute the node impact for each uncertainty, the pipeline must be run once without each uncertainty, and to compare the results, one additional time with all uncertainties.

Those last two measures are most important in the context of this work and should be understood well. For convenience, we will refer to μ_{var} with the "average variance" in the further work, although the correct term would be "average variance of the difference at the last stage".

Algorithm 3.5: Node Impact

Input: Current Pipeline Modules $\mathcal{M} = \{M_1, \ldots, M_p\}$, sample size n
Output: Results with one left out uncertainty each $\mathcal{R}$ and full-pipeline result R_{full}

$\mathcal{U} \leftarrow \emptyset$;
for *each module* $m_j \in \mathcal{M}$ **do**
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{\text{GetParameterUncertainties}(m_j)\}$;
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{\text{GetVisualUncertainties}(m_j)\}$;
end
$R_{full} \leftarrow \text{PipelineRun}(\mathcal{M}, \mathcal{U}, n)$;
$\mathcal{R} \leftarrow \emptyset$;
for *each* $u_k \in \mathcal{U}$ **do**
  $\mathcal{U}_{reduced} \leftarrow \mathcal{U} \setminus \{u_k\}$;
  $\mathcal{R}[u_k] \leftarrow \text{PipelineRun}(\mathcal{M}, \mathcal{U}_{reduced}, n)$;
end
return $\mathcal{R}, R_{full}$;

In case of a VU, the last four explained metrics are computed only on the area affected by the uncertainty, not the defined area of the uncertainty. This is because VUs have the peculiarity that apart from the impact on their respective area itself, they can have an effect on the outside of the region, with, for example, a blur after an uncertainty. Therefore, a VU can affect a bigger area (i.e. an envelope area) than the VU definition itself, which must be taken into consideration for calculations like the μ_{bias}, σ_{bias}, μ_{var}, and σ_{var}. Figure 3.5 shows an example where the definition shape (green) differs from the shape of the actual impact (orange) of a VU. The example was created with a pipeline with a *Circle Generator* with two VUs and a *Gaussian Filter*. In which the shape of the impact is computed by finding a contour around $|\mu_{diff}|$.

However, there may be cases in which multiple contours are found, especially when there are segmentation modules in the pipeline. In such a case, we compute the convex hull around all found contours. While this approach might not be completely accurate because each contour found should be treated independently, it is deemed sufficient as a proof of concept.

Node Impact Visualization

This section shows how the node impact is presented in the framework. A simple example is presented in Figure 3.6. The underlying pipeline consists of a *Circle Generator* and a *Gaussian Filter*. The reference value for the σ parameter of the *Gaussian Filter* is 5. Furthermore, one PU for the σ parameter of the *Gaussian Filter* is added with its parameters set to $\mu_0 = 0$, $s_\mu = 0$, $s_n = 5$, $t = uniform$. When inspecting the *Node Details (A)* after computing the node impact, both views of Figure 3.6 are available. The upper view is called *Impact Summary*, and the lower view is called *Impact Details (B)*. In both views at the top, the standard *Node Details (C)* are displayed, which describe

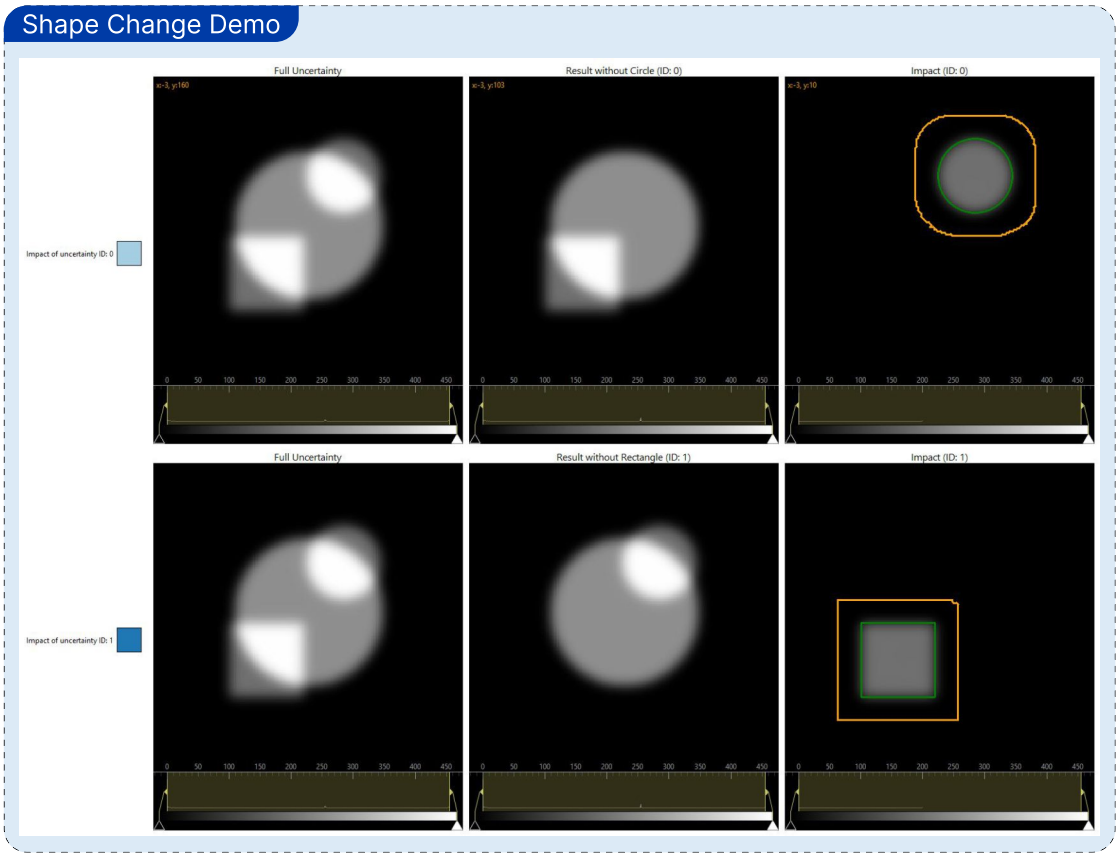


Figure 3.5: Demonstration on how the effect of a VU can transform the original shape. The demo shows a base circle with two VUs, one circular VU in the upper right and one rectangular VU in the lower left, that are passed through a *Gaussian Filter*. The three images in the first row show, from left to right, the image with both (all) uncertainties, the result without circular uncertainty, and the impact of the circular uncertainty. The second row represents the same, but for the rectangular uncertainty. The green line shows the shape of how the uncertainties have been defined, while the orange shape represents a contour around the actual affected area by the uncertainty.

the pipeline change of the current node. In the *Impact Summary (A)*, at first, only the main plot (D) is displayed, which shows the previously explained μ_{var} (202.46) and σ_{var} (418.62) for each present uncertainty. The user can then choose an uncertainty by selecting one from the main plot (D) to analyze it further; thus, showing the two images on the right (E) for the selected uncertainty, which visualize the μ_{diff} and the σ^2_{diff} of the last module. On the bottom, histograms visualize the corresponding distributions (F). The text in the middle displays details about the selected node in the main plot (D), which are the total average variance and the total average standard deviation. In the case of only one uncertainty, the measures are equal. Above the main plot (D), a user can determine what to visualize by choosing between the average variance, average standard deviation, and the bias. The other plots adjust accordingly to the selection made.

The *Impact Details*, located in the lower part of the figure, display the pipeline results details. On the left, the respective uncertainty is specified with the ID and the color code, followed by dropdowns for the module index and view mode, respectively. The module index specifies which module result shall be shown. In the example, the possible indices are 0 for the *Circle Generator* and 1 for the *Gaussian Filter*. The view mode specifies if the mean, standard deviation, variance, upper confidence interval bound, or lower confidence interval bound shall be visualized. More on the right are the respective visualizations, first with full uncertainty, then without uncertainty ID: 0, and the impact (difference/bias, depending on what the user is viewing). Note that the bias difference visualization in the *Impact Summary* part represents the same underlying data as the impact visualization in the *Node Details*, but with another LUT. The LUT maps the intensity values to color values, and can be seen below each image. To be comparable, images in the detailed view of the same row share the same LUT.

To identify uncertainties, the framework provides an integer identifier and a unique color. The colors used for that are from ColorBrewer2 [BHP], which is based on the work of Harrower et al. [HB03] that proposes color maps for different use cases. To colorize the uncertainties, we adapted the color map *12-class Paired*, which can be seen in Figure 3.7.

3.2.3 Change Types

This section describes which changes of the pipeline are captured and used to build the provenance. For better structure, we divide the changes into three levels: pipeline level, module level, and uncertainty level.

- *Start (Pipeline Level)*: A special case to represent the origin of the pipeline.
- *Add Module (Pipeline Level)*: A module is added to the pipeline.
- *Remove Module (Pipeline Level)*: A module is removed from the pipeline.
- *Change Resample Amount (Pipeline Level)*: The pipeline's resample amount has been changed.

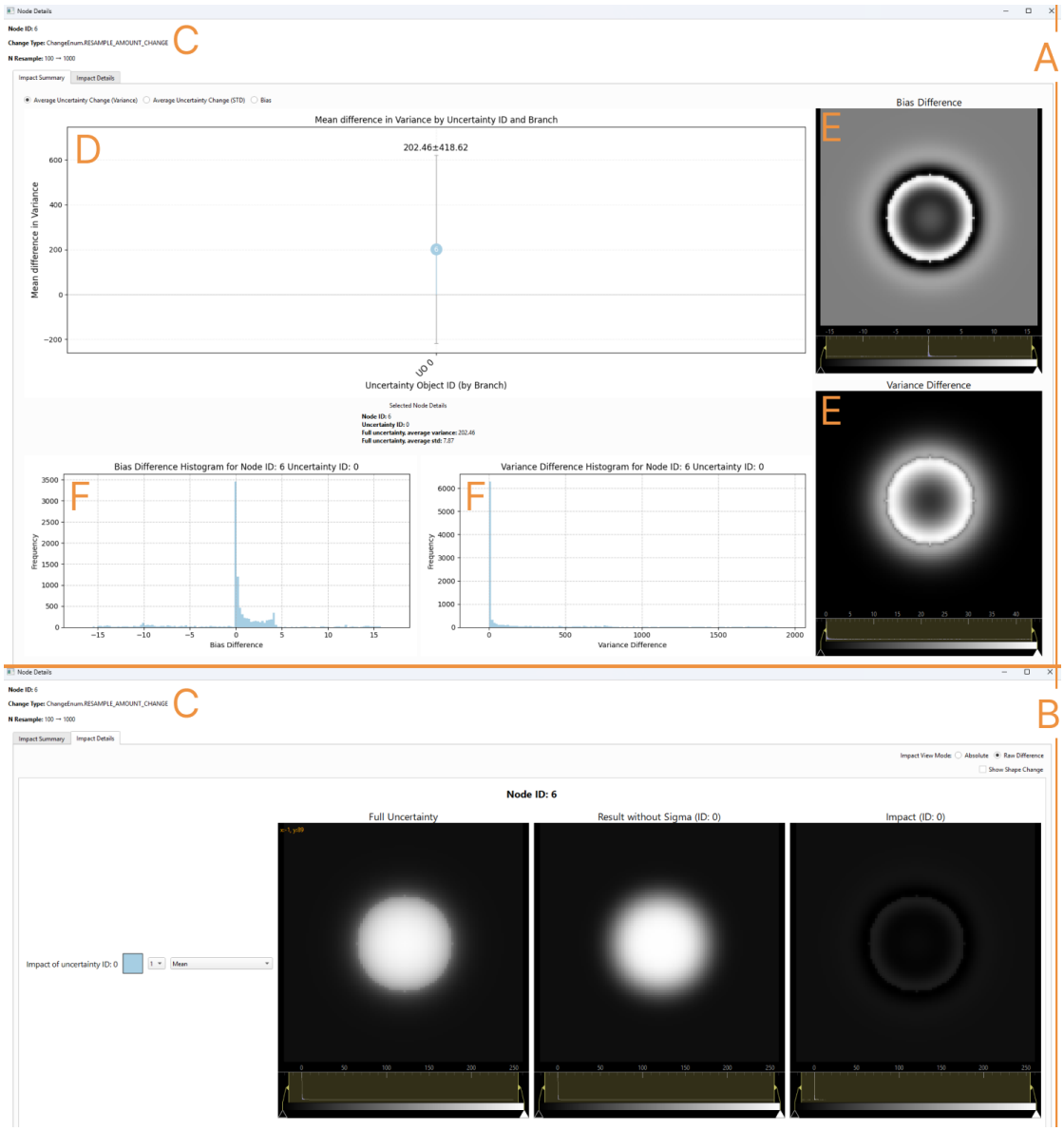


Figure 3.6: Shows how the *Node Impact* is presented in the framework. It consists of two main parts: the *Impact Summary (A)* (upper part) and the *Impact Details (B)* (lower part). In both views at the top, the change of the node that is visualized is displayed. The upper part of the figure displays an interactive plot, where the user can select an uncertainty that shall be analyzed further. The *Node Details (C)* offer the user to explore the different stages in detail and compare the bias, standard deviation, variance, or confidence interval between the uncertainty-excluded pipeline runs and the baseline.

Figure 3.7: Color map from Brewer et al. [BHP] and Harrower et al. [HB03]. The color map is conceptualized for qualitative data with 12 classes and is called the *12-class Paired* map.

- *Change Processing Unit Parameter (Module Level)*: A parameter from the module's processing unit has been changed.
- *Change Processing Unit Type (Module Level)*: The module's processing unit type has been changed.
- *Add Uncertainty (Module Level)*: Uncertainty has been added to the module.
- *Remove Uncertainty (Module Level)*: Uncertainty has been removed from the module
- *Change Geometry (Uncertainty Level)*: The uncertainty model's geometry, either size or position, has been changed.
- *Change Uncertainty Parameter (Uncertainty Level)*: A parameter from the uncertainty model has been changed.

3.3 Provenance

This section describes *Part B* of Figure 3.1. The framework captures all changes from the pipeline as provenance, enabling navigation between pipeline states, inspection of pipeline changes, and comparison between node impacts.

3.3.1 Provenance Visualization

To comprehend the lineage of uncertainty, we utilize provenance visualization. In literature, there are many ways to visualize provenance [XOW+20]. Although other forms, such as the sequence, might have been possible, we chose to visualize provenance as a hierarchical graph due to its intuitive representation of order and, therefore, navigation. Furthermore, the usefulness of a visualization ultimately depends on the actual implementation rather than its broad category. One disadvantage of the graph solution is that most usages of the framework will likely consist of single or at least long branches, which might not be a strength of a graph, but could be solved with grouping or collapsing strategies.

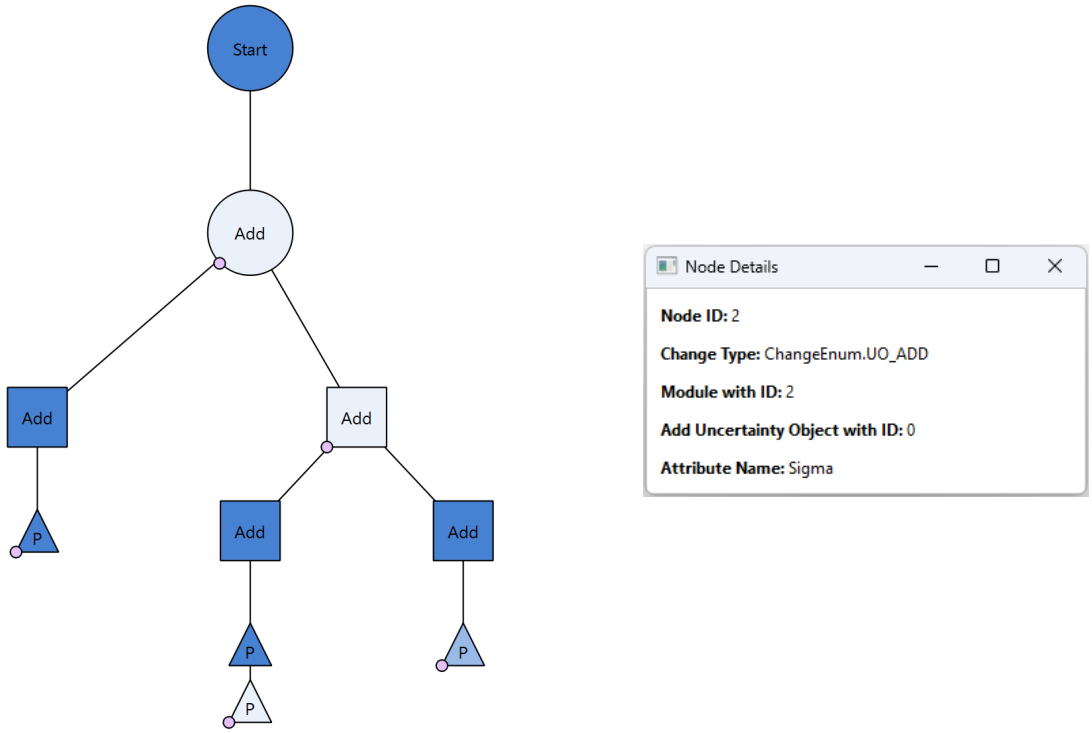


Figure 3.8: *Left side:* Shows an example provenance graph. The geometrical form indicates the level of change: (Pipeline Level - Circle), (Module Level - Square), and (Uncertainty Level - Triangle). The color ■ marks the current node, and therefore the currently loaded pipeline state, the brighter color ■ indicates selected nodes, and the ■ represents the default nodes. If the node impact for a node is computed, it is marked with a dot colored in ■. *Right side:* shows an example of *Node Details* for a node that did not have the node impact computed.

Figure 3.8 acts as a visual explanation for the design of the actual provenance graph, which will be explained in the following paragraphs. The previously explained change levels are displayed as follows:

- *Pipeline Level - Circle*: Changes at the pipeline level are visualized by a big circle.
- *Module Level - Square*: Changes at the module level are represented by a medium-sized square.
- *Uncertainty Level - Triangle*: Changes at the uncertainty model level are shown by a small triangle.

Moreover, each change has at its center a change abbreviation. To accurately represent the building process of the pipeline, we utilize a hierarchical graph layout that grows

downwards. Additionally, the nodes are spaced depending on the change level of the node. While higher levels are spaced farther apart, lower levels are spaced closer together.

When a user selects a node in the provenance graph, the pipeline will be reloaded to that state, and the new current provenance node will be marked ■. The node impact can be computed for the current node, and if processed, will be stored in the provenance node. Nodes with computed node impact are visually annotated ■.

If the user opens the details of a provenance node, they can see the pipeline change, as well as the node impact if it has been computed (see Figure 3.6). Furthermore, hovering over an uncertainty in the graph highlights the corresponding provenance path from the origin of the uncertainty to the inspected node.

Node Impact Comparison and Uncertainty Lineage

Analyzing a single node impact on its own represents uncertainty propagation, which quantifies the uncertainty at the output of the pipeline. Whereas the comparison of multiple node impacts is a type of parameter sensitivity analysis, because it investigates how the impact of uncertainties changes throughout specific pipeline changes. In that manner, a user can exploit the parameter space of the pipeline and uncertainty to gain insights into the sensitivity of its parameters. The downside is that this parameter sensitivity analysis method is by no means complete, because only selected parameter changes are investigated. More sophisticated techniques are much more computationally expensive, but they would explore the whole multidimensional parameter space. Thus, the method of simple comparison of node impacts might not detect dependencies or interaction effects since it is not a complete global parameter space analysis method [SAB+19]. Global parameter sensitivity analysis methods would yield better results for the user to interpret, but are not suitable for the proof of concept due to their high computational effort.

The pipeline view in Figure 3.9, the provenance view in Figure 3.10, the comparison summary view in Figure 3.11, and the comparison detail view in Figure 3.12 demonstrate an example for the comparison of node impacts. The example consists of two modules, a *Circle Generator* (Figure 3.9 A) and a *Gaussian Filter* (Figure 3.9 B), with a PU (Figure 3.9 C) introduced for the radius of the circle and a rectangular VU (Figure 3.9 D) in the same module. We compare four node impacts (Figure 3.10 E) with different values for σ (0, 1, 2, and 5) of the *Gaussian Filter* (Figure 3.9 A). Those changes with the respective node IDs can be seen in the comparison view at the top (Figure 3.11 F). The provenance view shows that the node impacts compared are partially on different branches (Figure 3.10 E), which is reflected in the comparison view in the main plot (Figure 3.11 G). There, the average standard deviations are first grouped by branch and then by the uncertainty. Branch 0 is the left, branch 1 is the middle, and branch 2 is the right. A user can select the different instances of uncertainty in the plot and analyze them, or hover to see the provenance path or the node. The detail view displays all uncertainties and their corresponding impact for each node (Figure 3.12).

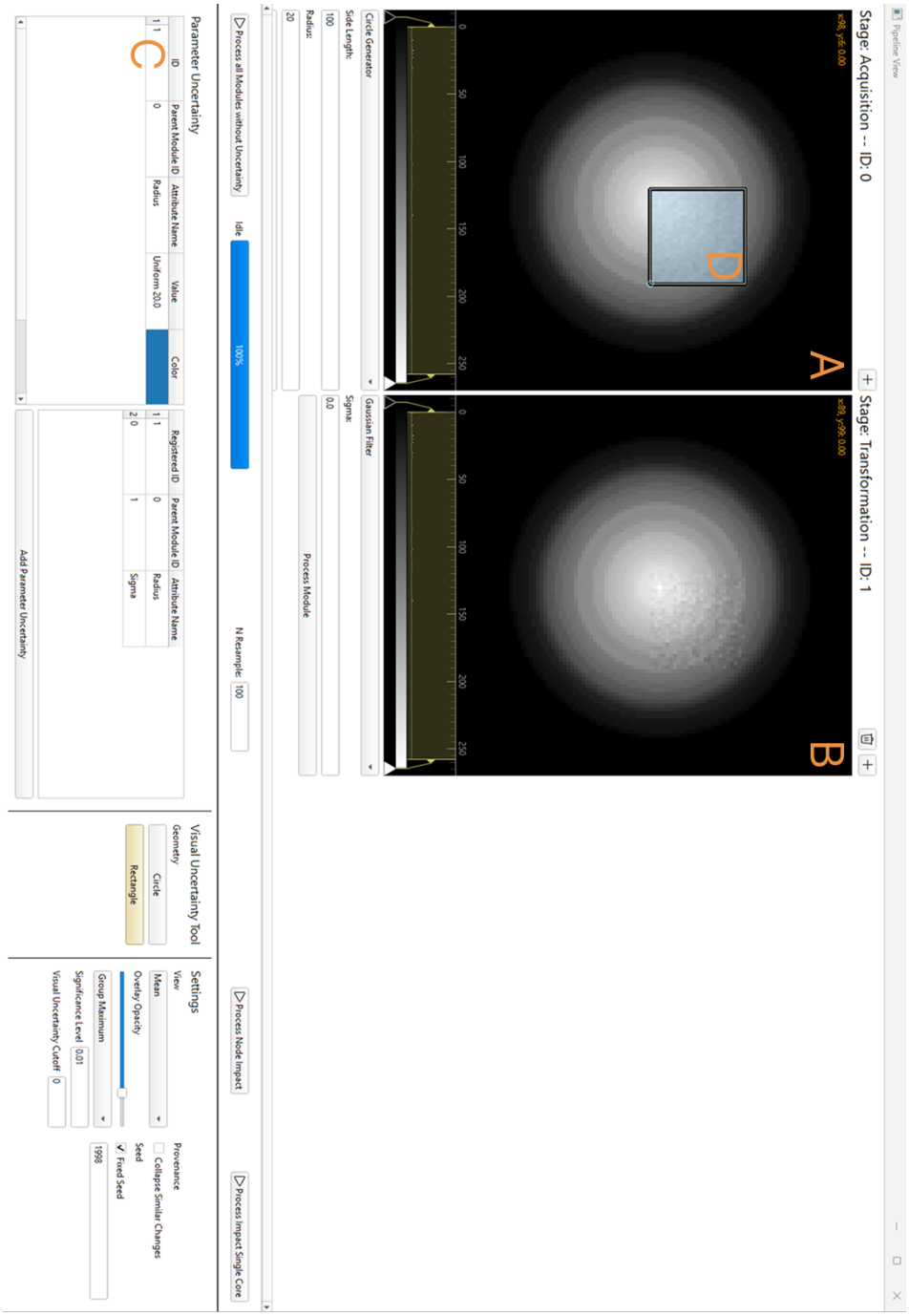


Figure 3.9: Base pipeline setup for the node impact comparison demonstration. It consists of a *Circle Generator* and a *Gaussian Filter*. Furthermore, the former includes a PU for the radius and a rectangular VU. The other nodes in the comparison only differ in the value of σ.

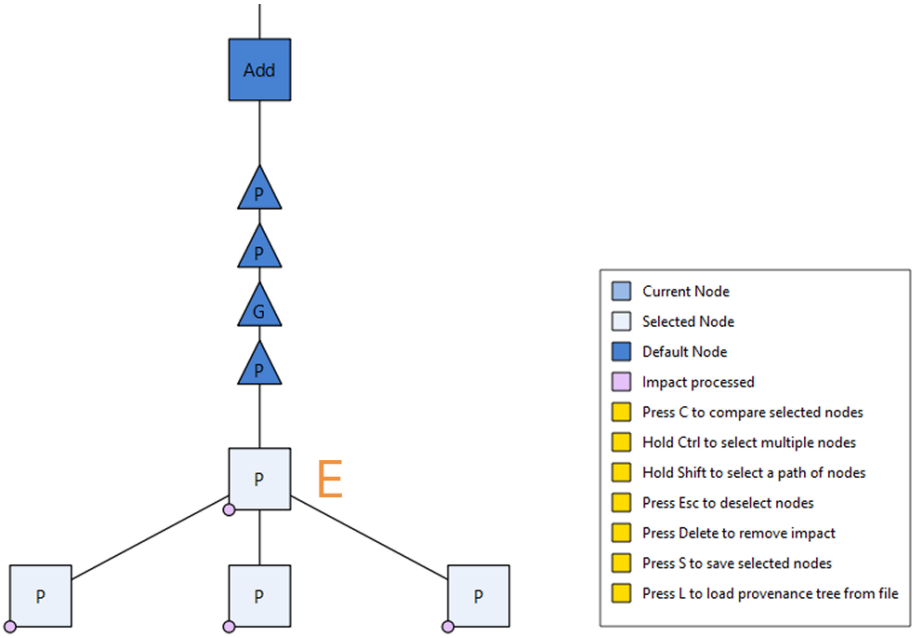


Figure 3.10: Provenance graph corresponding to the node impact comparison demonstration. The selected four nodes for which the node impact is computed are compared. The legend on the right describes the color code of provenance nodes and additionally presents the possible actions for a user.

3.4 Interface and Supported Analytical Process

Until now, each methodological part has been explained and discussed in detail. This section presents the framework as a whole, combining all its parts and focusing on user interaction. Furthermore, it describes the data flows present that are illustrated in Part C of Figure 3.1. The other examples used in the following sections serve no further purpose and are solely built to demonstrate the framework. First, we will explain the pipeline view, followed by the provenance view. The interfaces can be seen in Figure 3.13 and Figure 3.14.

3.4.1 Modules

The view of the pipeline is shown in the upper half, with the modules of the current pipeline. At the top of each module, there is a button to remove the module and a button to add a subsequent ■ transformation stage module. The exception is the ■ acquisition stage module that can not be removed. The central part of the module is the canvas, which visualizes the current output of the module and supports pan and zoom actions. In addition, each module shows the corresponding image coordinates and intensities of a hovering cursor. The LUT is positioned directly below the canvas and is followed by

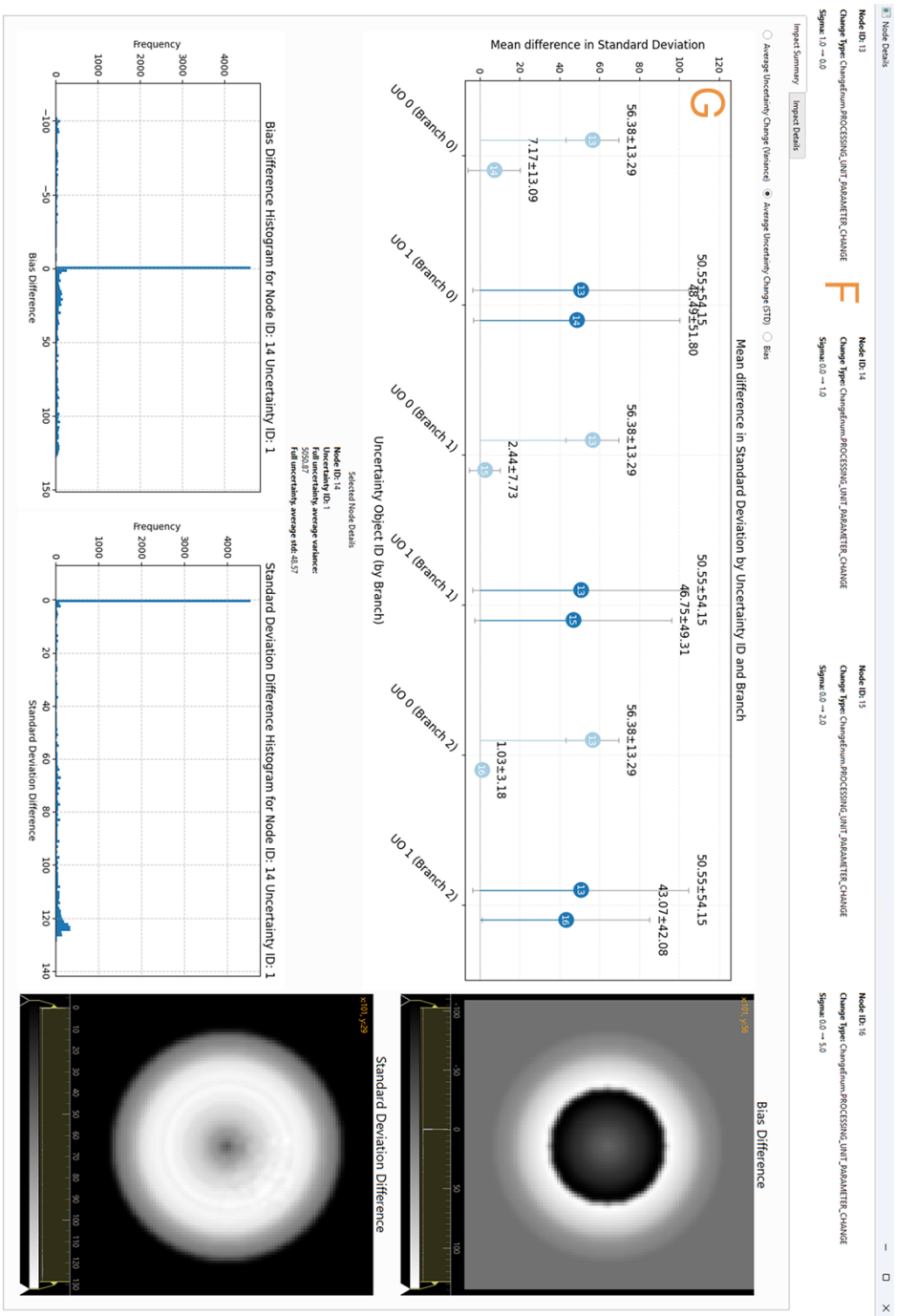


Figure 3.11: *Impact Summary* for the comparison demonstration. In the main plot, the uncertainties present are first grouped by the branch they are on and then by the uncertainty itself. The selected node in this example is uncertainty 1 at Node 14.

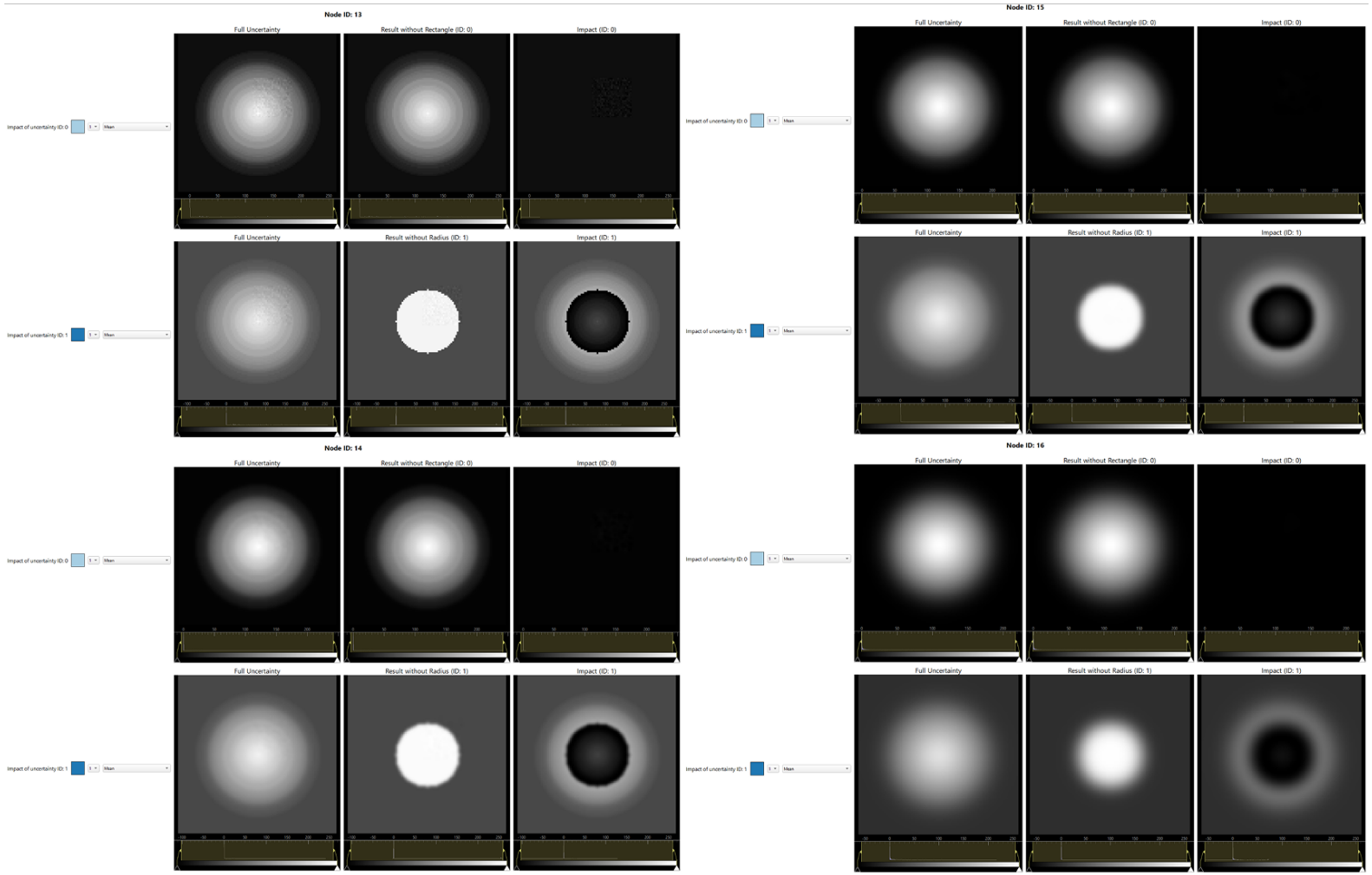


Figure 3.12: Corresponding *Impact Details* for the node impact comparison demonstration. Here, for each node, all its uncertainties that are present in the current pipeline state are listed. Each row can then be further inspected as previously explained.

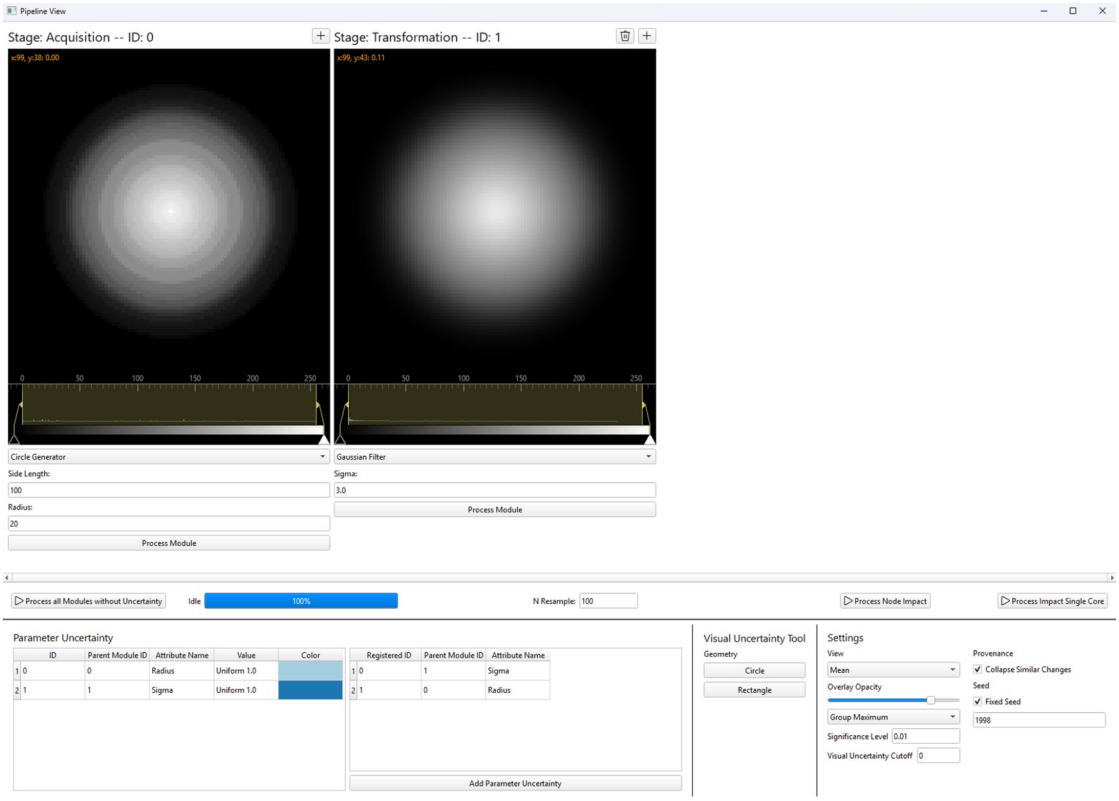


Figure 3.13: Demonstration of the interface from the pipeline view. In the upper half, the current modules with their parameters are displayed. Below each module is a button to process all preceding until the clicked one. In the middle section, there is the button to run the pipeline without uncertainty, the progress bar, the sample size n, and two buttons to compute the node impact for the current pipeline setting (once parallelized and once single core). On the bottom, the first table on the left lists the PUs that have been added. The table to the right shows the available parameters for which uncertainty can be introduced. Next to the tables is the form selector for the VUs, followed by the settings section.

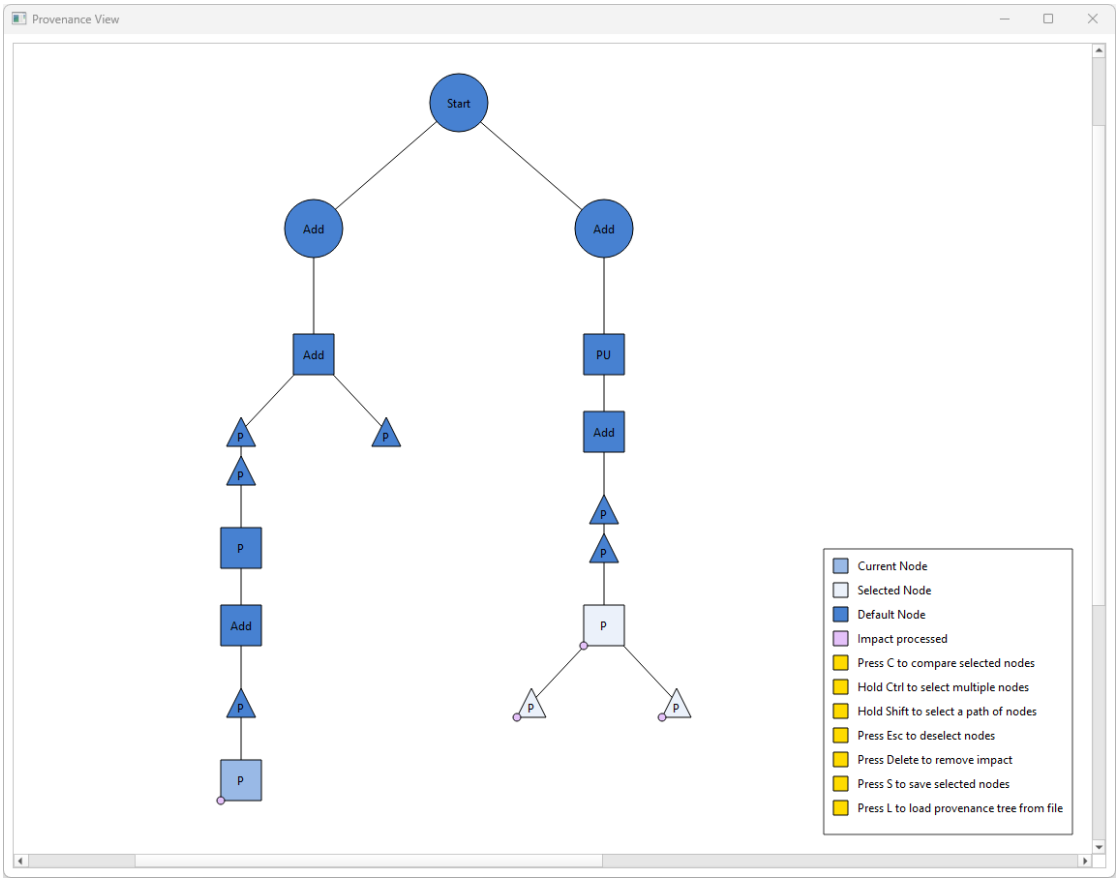


Figure 3.14: Demonstration of the provenance interface. The provenance is visualized as a hierarchical graph, and each node represents a pipeline state and a specific change itself. On the right side is the legend, which guides the different node colors and possible actions that a user can take.

a dropdown to select the processing unit of the module. The processing unit chosen determines the parameters displayed in the interface below.

3.4.2 Parameter Uncertainty

The *Parameter Uncertainty* section of the framework includes two interactive tables. The table on the right shows the available parameters for uncertainty propagation. In contrast, the left table shows the PUs that have been added to the pipeline already. Each row in the tables references the parent module of the parameter in addition to its identifier. To add a PU, first an available parameter has to be selected and then added with the button *Add Parameter Uncertainty*.

3.4.3 Visual Uncertainty

The section *Visual Uncertainty Tool* represents the selection of the geometry for the VU. A user can choose the geometry and add a VU by double-clicking at the center location of the desired position for the VU. The created uncertainties can be further moved by drag and drop, and resized with the handle on the outline of the geometry.

To edit PUs or VUs, a user has to right-click either a row of the table or a geometrical form in the module. This opens an edit dialog for the respective uncertainty model with all discussed parameters.

3.4.4 Processing a Pipeline

A user has multiple options to run a pipeline. Each module can be run separately with the *Process Module* button, which processes all preceding modules to the current module with all their currently defined uncertainties. For fast testing and pipeline building, a user can also process the whole pipeline without uncertainty with the *Process all Modules without Uncertainty*. To analyze the node impact of a current pipeline setting, a user utilizes the *Process Node Impact* button. When pressed, the entire pipeline is processed with all currently defined uncertainties, and it furthermore computes the impact of each uncertainty, which is then passed as the node impact to the provenance graph. Additionally, the framework provides a bar to show the progress of the current processing task, because processing potentially may take some time, depending on the data and sample sizes.

3.4.5 Settings

The settings section defines settings related to various parts of the framework, which are explained in the following paragraph.

View Settings

The first dropdown in this part controls what is displayed in the pipeline above. A user can switch between the *Mean*, *Standard Deviation*, and *Variance*. Furthermore, it is possible to add the standard deviation or variance as an overlay, and control its opacity with the slider below.

LUT Mode

A user can switch between three LUT modes: *Manual*, *Individual*, and *Group Maximum*. In *Manual* mode, the framework does not change the LUT settings on its own. A user can thus set the LUT interval manually for each view separately or apply a general LUT to all views with the help of the two input fields below. In *Individual* mode, the settings are optimized for each visualization view independently. Lastly, in *Group Maximum* mode, the framework adjusts related views to the minimum and maximum of all affected views, which keeps them visually comparable.

Significance Level (α)

The value of this control defines α of the confidence intervals described previously in the part *Evaluation and Statistics* of Section 11.

Visual Uncertainty Cutoff

As mentioned, VUs can have effects outside their defined region. However, this effect sometimes has a negligible effect on some pixels, due to the effect fading out, for example. Therefore, a user can set a cutoff threshold that defines at what order of magnitude a change is considered to be an influence.

Collapse Similar Nodes

While rapidly building a pipeline, a user often wants to manually test parameters and, therefore, may not want to track every single change. Therefore, there is the option to collapse the changes of the same kind on the same object or parameter if they are direct neighbors. Collapsible change types are: *Change Resample Amount*, *Change Processing Unit Parameter*, *Change Geometry*, and *Change Uncertainty Parameter*.

Seeding

Since Monte Carlo methods are based on sampling, which relies on randomness, it is necessary that the user can take control of the seed that initializes any random number generator. The user has the option to fix the seed to an arbitrary number or to leave it unfixed, resulting in varying results from the same inputs. If the seed is not fixed, the framework must still internally generate a temporary random seed to obtain comparable results. Remember that to determine the effect of the impact of an uncertainty X and Y, the framework must process the pipeline with both uncertainties, once with X and once with Y. To stay comparable, the seed must be set to the same value for each of the three pipeline runs. For that reason, the seed is reset before each sampling procedure.

Provenance View

The provenance graph is a canvas that displays the provenance graph. Changes to the pipeline add nodes to the graph. As discussed, the provenance data is presented as a tree in different geometrical forms and colors based on their level of change and selection state. To navigate the graph, the canvas supports zoom and pan actions. A left click on a desired node initiates a switch to another pipeline state and marks that node as the current node. A user can utilize this mechanic to explore pipeline states rapidly. After each switch, the pipeline is immediately processed without uncertainty to provide some orientation. A right click on a node opens its details, which displays the change the node represents and, if present, the computed node impact. To select nodes, a user can use the common pattern to select items by holding either *Ctrl* for single selection and *Shift* for chain selection in combination with left-clicks. A selection can be canceled by pressing *Esc*. While pressing C on a selection initiates a comparison of node impacts, pressing *Del* removes the stored node impacts on the selected nodes, and pressing S saves the selection of provenance graph to a file. The file can then be loaded with L in the provenance view. Note that pressing S when no nodes are selected enables a user to save

the whole provenance graph. All hotkeys and colors of nodes are shown and explained in a legend on the provenance graph canvas.

3.4.6 Data Flows

To summarize, we will refer to *Part C* of Figure 3.1. *Pipeline Changes* are tracked in the pipeline view and stored in the provenance graph at the respective node. The same applies to the *Node Impacts*. The two main purposes of the provenance tree are the navigation between *Pipeline States* and the comparison of the *Node Impacts* to gain insights into the uncertainty propagation and uncertainty lineage.

CHAPTER 4

Implementation

This chapter lays out the concepts, technologies, and environments used to implement the framework, while reasoning for their selection. As the goal is to implement a framework for proof of concept, the framework itself should be built with languages and third-party frameworks that are well-known in the scientific community. Thus, the framework is easily accessible for scientists who try to understand the core concept or aim to extend its functionality.

4.1 Preliminaries

While brainstorming how to implement the framework at its core functionality, extensive experimentation has been conducted in Jupyter Notebooks [KRKP+16]. Initially, GPs have been the central point of attention in combination with the GPvecchia R package from Katzfuss et al. [KG21]. As we realized that using GPs for our work is out of scope, we continued with the simpler Monte Carlo methodology and tested it again in Jupyter Notebooks.

4.2 Languages and Frameworks

Python [Pyt23] is widely used by the scientific community, as it is comparatively easy and, combined with available powerful packages, it is a good choice for fast prototyping. Thus, we chose to develop a Python desktop application, with PySide6 [The25], which is a Python wrapper for the cross-platform application development framework QT [Theb]. Furthermore, we utilize the GUI library PyQtGraph [Co14], which is a library for PySide applications. The data in the framework is loaded and passed as NumPy [HMvdW+20] array instances. Other packages the framework is using are listed in Table 4.1; furthermore, their use is depicted in Figure 4.1.

Package	Version
PySide6 [The25]	6.8.3
PyQtGraph [Co14]	0.13.7
NumPy [HMvdW+20]	2.2.6
Matplotlib [Hun07]	3.10.3
NetworkX [HSS08]	3.5
SciPy [VGO+20]	1.16.0
OpenCV-Python [Bra00]	4.12.0.88
scikit-image [WSNI+14]	0.25.2
Pydicom [Msm+24]	3.0.1
jsonpickle [Dev25]	4.1.1
psutil [Rod25]	7.0.0

Table 4.1: Python packages that the framework is using, with the respective version number

To build well-structured software, we use a custom version of the Model-View architecture from QT. In the architecture, each view has an underlying model, while the view defines the user interface, the model handles the domain logic. Thus, each view can directly interact with its model. Meanwhile, the model defines QT Signals, which any other view can connect to. In that way, we achieve separation of concerns between model and view and still have bidirectional communication.

The following sections will explain the main parts of the framework and their implementation in more detail.

4.2.1 Pipeline View

Almost every widget in the pipeline view user interface is a standard widget from PySide6. The only exceptions are each module's visualization view and its LUT widget, which we built with PyQtGraph. Under the hood, each module's visualization view is a ViewBox with an ImageItem for the base image and the overlay image, and if present, a PlotCurveItem for the contour. A module's model is its processing unit and defines the data model for its parameters and the domain logic. Thereby, we utilize existing implementations for some types of processing units. For the *Gaussian Filter*, *Median Filter*, and *Laplace Filter*, we utilize the corresponding methods from SciPy, the *Morphology Filter* is implemented with OpenCV-Python, and the *Otsu Threshold Segmentation* implementation uses scikit-image. Meanwhile, the *Threshold Segmentation* is implemented with NumPy on its own, and in the logic for the *Image Acquisition* processing unit, we utilize OpenCV-Python and Pydicom.

To track changes, a Python decorator is used to annotate the methods of interest with a specification of the change type and paths that the change applies to (see Listing 1).

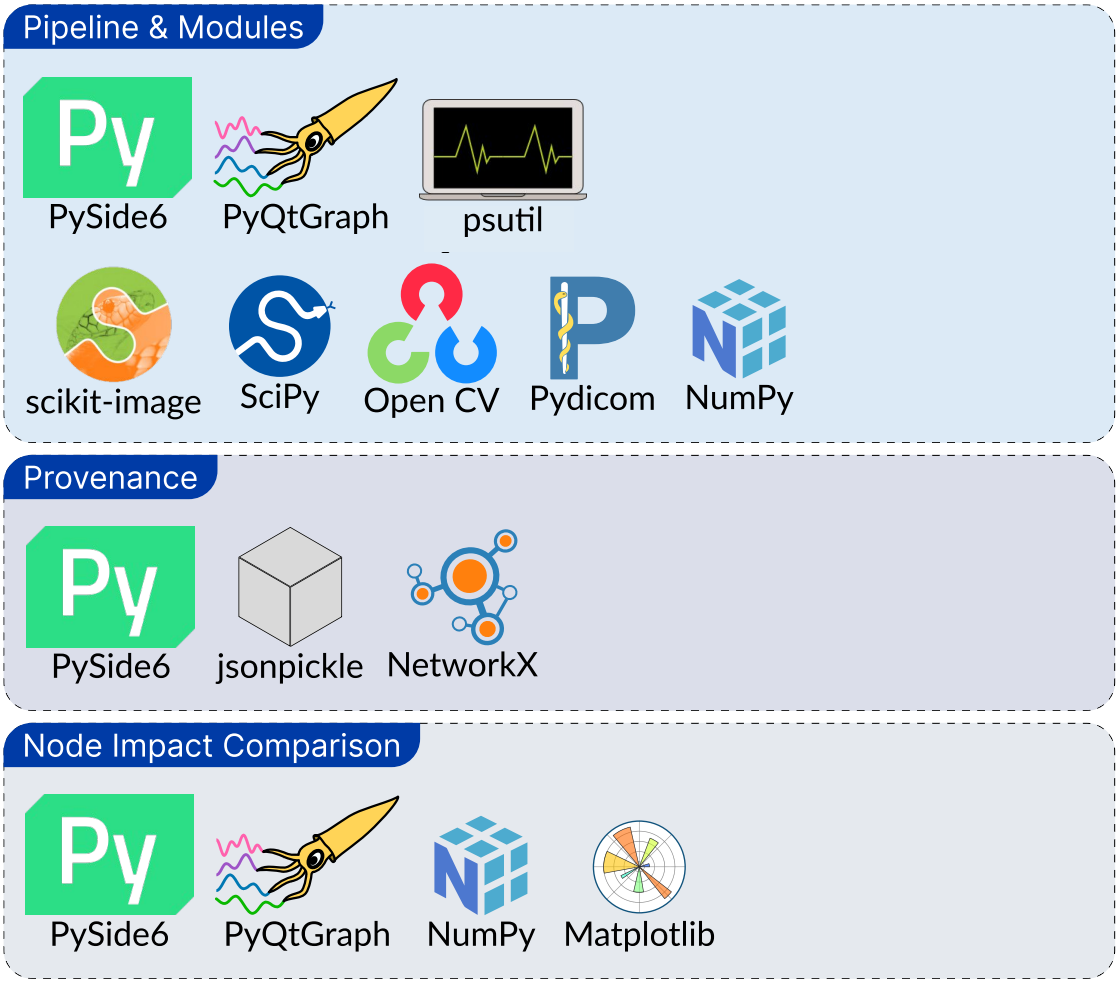


Figure 4.1: Illustrates which parts of the framework made use of which Python packages.

With the help of the EventBus, which is a custom singleton that defines a QT Signal, we can publish the recorded change to any consumer.

```
@track_change(change_kind=ChangeEnum.PROCESSING_UNIT_PARAMETER_CHANGE)
def set_sigma(self, sigma: float):
    self._sigma = sigma
    self.sig_sigma_changed.emit(sigma)
```

Listing 1: Track change decorator example. If one wants to track the changes of sigma, the only thing needed is to decorate the method and specify the type of change.

4.2.2 Provenance View

The interface of the provenance view is a customized form of the QGraphicsView from PySide6 and visualizes the underlying provenance graph of the model. That provenance model represents the provenance graph with a DiGraph from NetworkX. Furthermore, it implements a custom tree layout algorithm and defines the domain logic around it. As one major reason for provenance is reproducibility, a user can save the whole or parts of the provenance tree to a JSON file. The implementation of loading and saving the JSON file uses the jsonpickle package with some custom conversion handlers.

4.2.3 Node Impact View

The visualization of the node impact consists of similar visualization views as used for the modules, histograms, and the interactive main plot, built with Matplotlib.

4.3 Performance

High-resolution images, a high sample size, and multiple uncertainties together can lead to a massive effort to compute the node impact. For similar reasons, we decided that the proof of concept shall only handle 2D data so that we can focus on the uncertainty topic. Luckily, the runs performed in a Monte Carlo experiment are independent of each other, and thus parallelization is possible. The three main options investigated are *Multithreading*, *Multiprocessing*, and *GPU Computing*. *Multithreading* is not a real option in our case, as there are no I/O bound tasks that require waiting for specific events. *GPU Computing* may lead to promising speed-ups, but it would have been by far the biggest effort to implement. Thus, we implemented a simple form of *Multiprocessing*. When computing the node impact, the framework spawns processes or reuses an existing process pool. The number of processes depends on the number of CPU cores and, if the memory allows, spawns as many processes. Remember the three steps of the uncertainty propagation methodology from Section 3.2.2. The first step of preparing the pipeline parameter sets is performed on one core. Those sets are then split up into chunks and distributed across the process pool. Therefore, the second step of running the pipelines is executed in parallel. The last step of collecting and evaluating the results of the processes is again performed by one core. Some measured manual comparisons of the *Multiprocessing* approach against the default approach resulted in a speedup of 5 times. The processes used in the parallel execution were 27, which shows that the overhead of distributing the chunks and recombining their results does take a lot of resources. Parallelization in general is a complex topic, and there might be many options to improve and optimize the currently implemented *Multiprocessing* approach. Nevertheless, parallelization is not a central topic of this thesis, and the approach should only serve as a proof of concept and ease testing and result generation. Furthermore, the *GPU Computing* approach might yield even greater results.

4.4 Extendability

The framework is designed to be easily extendable by later developers, allowing them to add new modules and perform additional tasks. Furthermore, it is possible to unplug the Monte Carlo processing logic and plug in some other methodology. Extending the approach to 3D data is theoretically possible, but the scalability of the methodology to higher dimensions should be evaluated for complexity and thus, inherently, performance first.

4.5 Development

Various software was in use for the development. As mentioned first approaches were done in Jupyter Notebooks. The earliest version of the framework was built with CustomTkinter [Tom] and was soon dropped because of the low popularity, but mainly because PyQtGraph is built for PySide. Thus, the switch was inevitable. With the switch from CustomTkinter to PySide6 came also the switch from the IDE VS Code [Mic] to PyCharm [Jet24].

CHAPTER 5

Results

In the course of this chapter, we use the developed framework that implements the methodology described in Chapter 3 within *Usage Scenarios*, which demonstrate the potential usages of the framework on multiple datasets.

5.1 Evaluation

The evaluation of the framework follows the principles of Qualitative Result Inspection (QRI) form Isenberg et al. [IIJ+13]. The approach involves creating meaningful usage scenarios to test the effectiveness of the developed visualizations, which provides readers of the thesis with the means to assess the work independently. Additionally, we critically analyze whether and to what extent the aforementioned research questions have been addressed.

Usage scenarios aim to put a reader into the role of a user of the framework and try to encourage the reader to agree on the quality and validity of the statement [IIJ+13], by inspecting the results presented in the usage scenario. Isenberg et al. [IIJ+13] found two main approaches to perform QRI: comparative and isolated ones. Comparative QRI contrast against the result or solution of other state-of-the-art approaches. While isolated QRI lacks such comparable state-of-the-art approaches, and, therefore, needs a solid problem description as well as justification for the solution. The approach of this work is to be classified into the latter.

5.2 Datasets

As it is beneficial to know the ground truth in uncertainty propagation tasks, we utilize toy and phantom datasets. For phantom datasets, the ground truth of the locations of the objects of interest might also be known, but the information itself is much more vague

due to uncertainties in the phantom building and acquisition process. As the framework only supports two-dimensional data, it is advantageous if a slice of a phantom dataset includes multiple regions of interest to work with. Furthermore, not all datasets suit the framework, because they would require more sophisticated modules, e.g., segmentation tasks. The following paragraphs provide a detailed explanation of all the datasets used.

Toy-9-Circles Dataset

The Toy-9-Circles toy dataset is self-generated and consists of a background of intensity $I = 100$ and 9 circles in the foreground, whose intensities are equally distributed between $[100, 118]$. For clarification, the dataset is visible in Figure 5.1. It was generated to emulate lesions in a very simplistic way and allows for testing uncertainties as a first step.

Figure 5.1: The Toy-9-Circles dataset. The background has an intensity of 100. The circles have an incrementing intensity starting at 100, going up to 118.

QIBA-CT-Liver-Phantom

The dataset QIBA-CT-Liver-Phantom [ZLL+21] is from the Cancer Imaging Archive [Thea]. It includes three datasets itself, out of which we use only the "Dataset #3 Iterative Reconstruction phantom." This dataset is a liver phantom that has 10 inserted lesions of varying sizes and radiodensities. The dataset was acquired in multiple series using a CT scanner across multiple different acquisition setups and protocols, resulting in a set of 235 series. A slice of one of those series can be seen in Figure 5.2.

CT-Phantom4Radiomics

The CT-Phantom4Radiomics [SBO+23] is a phantom dataset from the Cancer Imaging Archive [Thea]. It consists of 238 CT series, with 8 parameter variations, and is built using paper potassium iodide solution paper-printing technique. The dataset shows

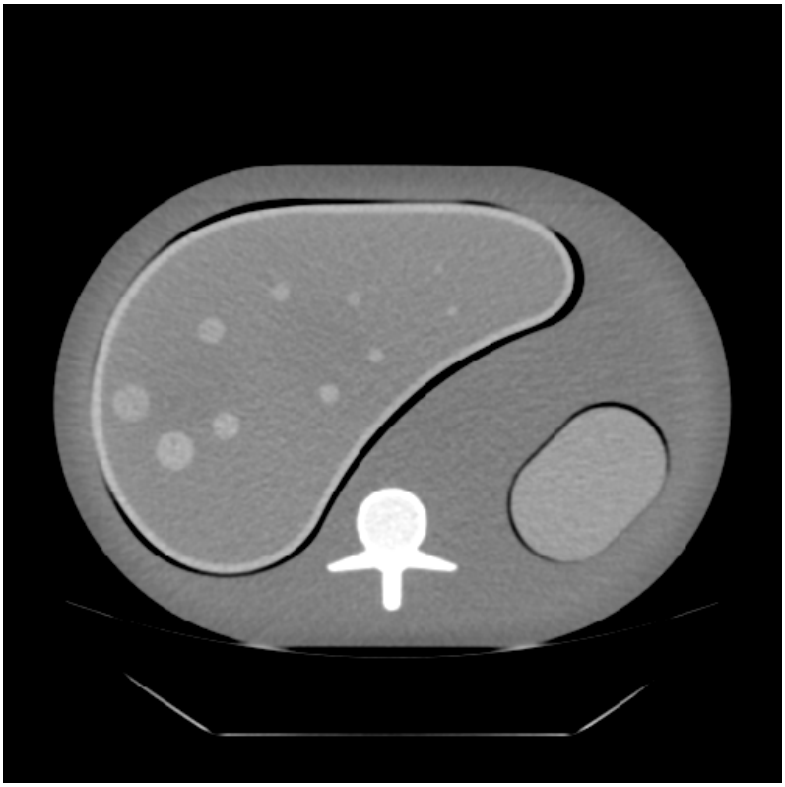


Figure 5.2: The 49th slice (file "1-49.dcm") of series number 8 from the third dataset of [ZLL+21]. The slice visualizes a phantom of the liver that has 10 inserted spherical lesions of varying size and radiodensity.

normal liver tissue as well as lesion tissue, which are both annotated. A slice of the dataset with highlighted segmentations is shown in Figure 5.3.

5.3 Usage Scenario 1

5.3.1 Scenario Definition

As described in Chapter 2, uncertainty may occur at each step of the medical visualization pipeline. As it is currently impossible to counteract all sources of uncertainty, especially aleatoric uncertainty, it is beneficial for the clinician to gain insights into the general effect of each module they are utilizing. In such a scenario, a user can set up a pipeline with a module of interest and analyze the module's effect on the uncertainty of their choice. Thus, with the gained awareness, a clinician may build or adjust a pipeline differently.

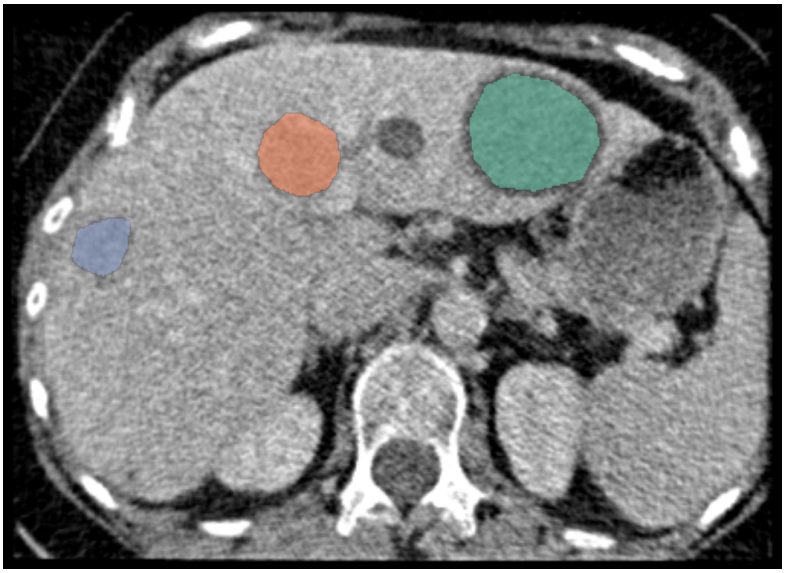


Figure 5.3: The 143th slice (file "1-200.dcm") of series number 414 from the data set CT-Phantom4Radiomics [SBO+23]. The creators of the dataset provide a ground truth segmentation for the lesions. Three of those are highlighted in the Figure.

5.3.2 Task 1.1 - Single Module Analysis

For that purpose, we analyze the uncertainty effects of all currently implemented modules on the same pipeline, compare and categorize them. To do so, we use different configurations of uncertainties and investigate if certain trends are evident.

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. Each ■ transformation stage module is used once
- Uncertainties
  1. VU with rectangular shape over the whole image.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 10, 50, 100$, $t = uniform$
- Sample size $n = 1000$

As a first step, all modules process the self-generated Toy-9-Circles toy dataset (see Figure 5.1). The uncertainty added is a rectangular VU that completely covers the image with a mean offset μ_0 of 0 and an absolute noise-spread s_n (10, 50, and 100) uniformly distributed. Then each ■ transformation stage module is tested with a sample size n of 1000, and their values of the mean standard deviation μ_σ are compared. The task is

performed three times with a different noise-spread setting for the VU each time.

Figure 5.4 shows the corresponding provenance graph and node details of Task 1.1. Figure 5.5 and Figure 5.6 show one node comparison plot for each noise-spread s_n setting. Since the provenance graph splits up from the parent node, each module is on a separate branch; thus, they are grouped in the plots with the same parent node ID 12 (see branching in Figure 5.4 below node ID 12). The plot in Figure 5.5 clearly shows that in this task, the *Gaussian Filter*, *Median Filter*, and both *Morphology Filter* operations (erosion and dilation) decrease the average standard deviation in both cases, whereas the *Laplace Filter*, *Threshold Segmentation*, and *Otsu Threshold Segmentation* increase the average standard deviation (compare change of the data-points of each module group). Since the tested uncertainty setting represents noise, it is expected that blur operations as the Gaussian and *Median Filter*, reduce the average standard deviation. For the same reason, the *Laplace Filter* and both *Threshold Segmentation* and *Otsu Threshold Segmentation* modules amplify the noise.

Noticeably, the increase in average standard deviation for all modules is proportional to the noise-spread s_n value, and thus, they show linear behavior (see Figure 5.5 and Figure 5.6 with the s_n specification in the upper left of each main plot). Let's take the *Gaussian Filter* as an example. In the main Figure 5.5 with $s_n = 10$, the pipeline with a *Gaussian Filter* has an average standard deviation of 0.82 (second data point). If we look into Figure 5.6, we see average standard deviation values for $s_n = 50$ and $s_n = 100$, which are 4.1 and 8.19. Thus, one can see that the scaling of those standard deviations matches the scaling of the magnitudes for the uncertainty parameter s_n in the *Gaussian Filter* case. Exceptions to this are the two threshold modules, which exhibit non-linear behavior. That is interesting because the *Median Filter* and the *Morphology Filter* operations are nonlinear filters [Uch13], but do show linear behavior in this case. Another unexpected thing is that both *Morphology Filter* operations (dilation and erosion) show exactly the same values (see Figure 5.5 and Figure 5.6).

5.3.3 Task 1.2 - Impact of Preceding Blur

With those results in mind, a user may investigate if one of the modules that has been shown to increase the average standard deviation performs better with a decreasing uncertainty module connected upstream. Thus, we extend this first scenario and test the three uncertainty-increasing modules (*Laplace Filter*, *Threshold Segmentation*, and *Otsu Threshold Segmentation*) once with a preceding *Gaussian Filter* and once with a *Median Filter*.

Figure 5.4: Node details and provenance graph of Task 1.1 - Single module analysis. The graph shows that the node impact is computed for all leaf nodes and node 12, which are compared further in the task.

Task 1.1 - Comparison Plot 1/2

Node ID: 12

Change Type: ChangeEnum.RESAMPLE_AMOUNT_CHANGE

N Resample: 100 → 1000

Node ID: 14

Change Type: ChangeEnum.PROCESSING_UNIT_PARAMETER_CHANGE

Sigma: 1.0 → 2.0

Node ID: 15

Change Type: ChangeEnum.PROCESSOR_UNIT_CHANGE

Module with ID: 1

Processor Unit: Gaussian Filter → Median Filter

Node ID: 16

Change Type: ChangeEnum.PROCESSOR_UNIT_CHANGE

Module with ID: 1

Processor Unit: Gaussian Filter → Laplace Filter

Node ID: 17

Change Type: ChangeEnum.PROCESSOR_UNIT_CHANGE

Module with ID: 1

Processor Unit: Gaussian Filter → Morphology Filter

Node ID: 19

Change Type: ChangeEnum.PROCESSING_UNIT_PARAMETER_CHANGE

Morphology Type: MorphologyTypeEnum.DILATION → MorphologyTypeEnum.EROSION

Node ID: 21

Change Type: ChangeEnum.PROCESSING_UNIT_PARAMETER_CHANGE

Lower Threshold: 0 → 100.0

Node ID: 22

Change Type: ChangeEnum.PROCESSOR_UNIT_CHANGE

Module with ID: 1

Processor Unit: Gaussian Filter → Otsu Threshold Segmentation

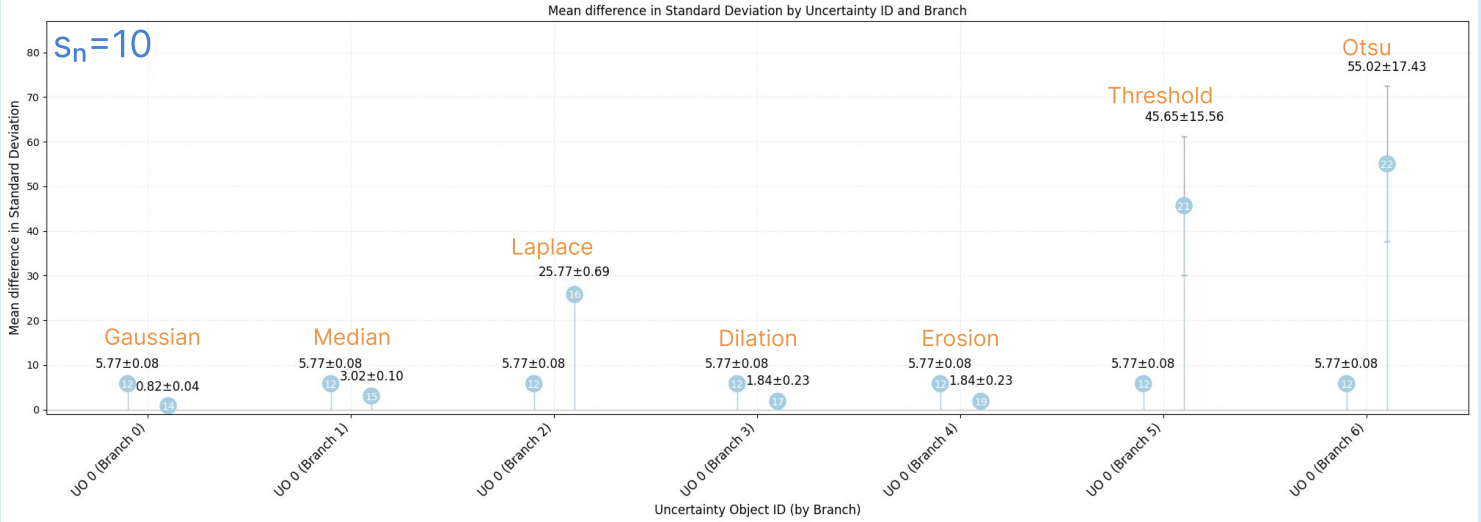


Figure 5.5: Shows the first part of the comparison plot of Task 1.1, which includes the node details and the comparison plot for $s_n = 10$. Each node is grouped with the parent node 12. Thus, in each group, the left data point shows the parent value of the μ_σ and the right shows the value with the respective module connected. The plot reveals the increase of the average standard deviation with the addition of a *Laplace Filter*, *Threshold Segmentation*, or *Otsu Threshold Segmentation*. All other modules decreased the average standard deviation.

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. Each blur module (*Gaussian Filter*, *Median Filter*), one time
  3. Each uncertainty-increasing module from Task 1.1, one time (*Laplace Filter*, *Threshold Segmentation*, *Otsu Threshold Segmentation*)
- Uncertainties
  1. Rectangular Uncertainty over the whole image.
     $\mu_0 = 0$, $s_\mu = 0$ $s_n = 50$ $t = uniform$
- Sample size $n = 1000$

The results are presented in Figure 5.7 and Figure 5.8. The comparison plot Figure 5.7 shows a plot for the preceding Gaussian and *Median Filter*. In each plot, the values are grouped per uncertainty-increasing (*Laplace Filter*, *Threshold Segmentation*, and *Otsu Threshold Segmentation*). Such a group consists of four data points, where the first point corresponds to the average standard deviation in the pipeline with only the acquisition module and the rectangular VU. The second is the measure with the respective blur module. The third data point is the measure by which the particular uncertainty-increasing module is connected. Lastly, the fourth data point visualizes the measure with the blur removed again and therefore is equal to the results in Task 1.1. Notably, the *Laplace Filter*, which in Task 1.1 increased the average standard deviation, now decreases the uncertainty from the preceding *Gaussian Filter* (see Plot 1, Group 1, Decrease from data point 2 to 3). In all other cases, the modules still increase the average standard deviation from the blur modules. Nevertheless, in every case, the average standard deviation is lower than without the preceding blur modules.

Figure 5.8 compares the resulting image from the pipelines of Task 1.2. The different images are placed on a grid. Note that the Gaussian column can not be compared directly to the Median column, because they are different operations, and therefore, they can not be set to the same intensity. But one can compare both of them to the column without a preceding blur module. The *Gaussian Filter* improves the contrast for all three tested modules. While the *Median Filter* enhances the contrast for the segmentation modules,

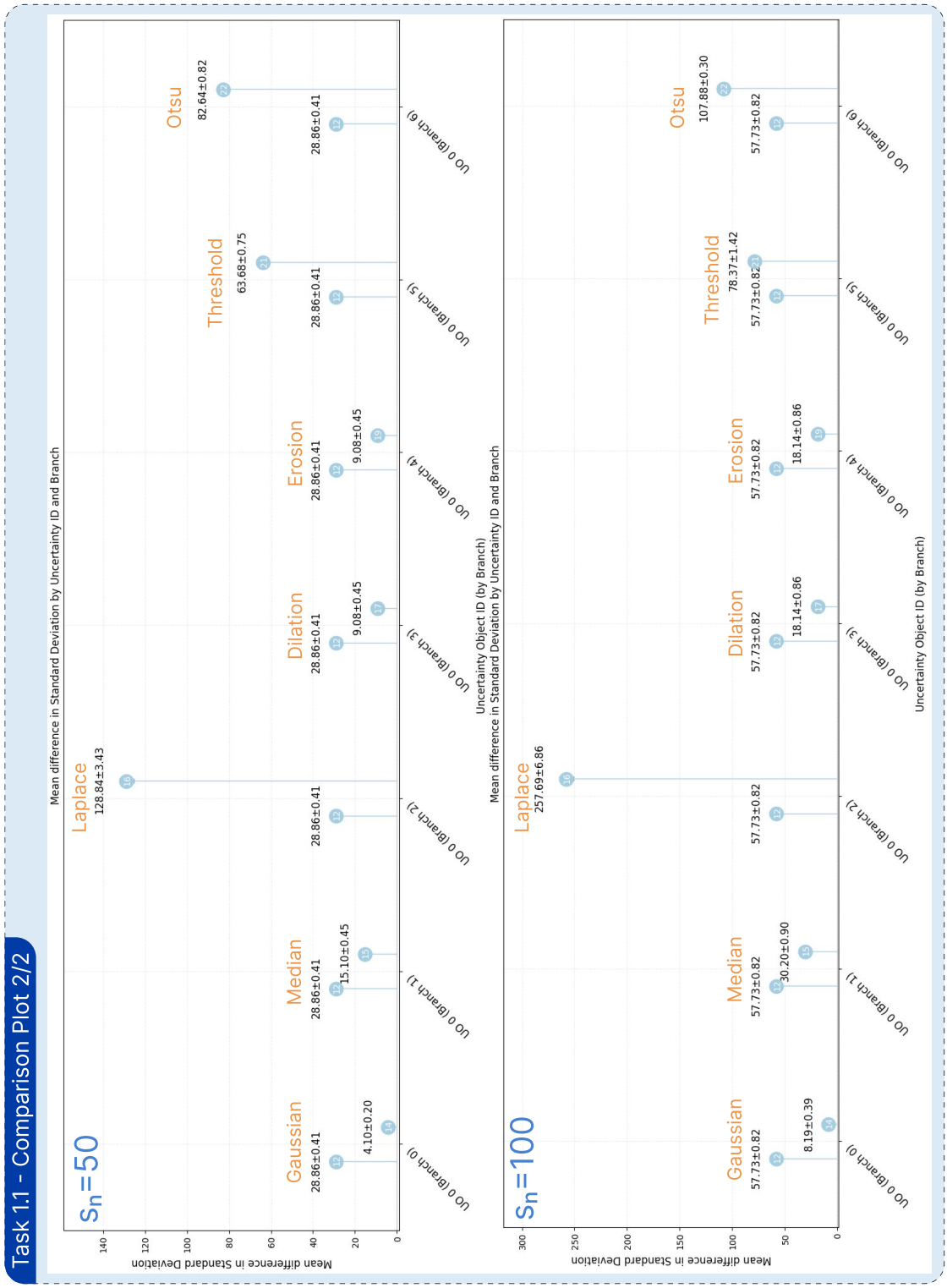


Figure 5.6: Extension to Figure 5.5 with the last two executions of Task 1.1 with $s = 50$ and $s = 100$. Notably, the same modules increased or decreased the average uncertainty, respectively, as in the first execution with $s = 10$.

it decreases the contrast slightly for the *Laplace Filter*.

The results presented in Tasks 1.1 and 1.2 show how a clinician or user can analyze modules, compare, and thus investigate their impact on uncertainty. In that manner, one can gain knowledge on how to build a pipeline that is less prone to uncertainties and therefore less prone to errors.

5.4 Usage Scenario 2

5.4.1 Scenario Definition

It is a major task for a clinician to detect abnormalities or diseases in scans, as uncertainties might cause visual artifacts or even hide elements of interest from the eyes of an expert. Especially in segmentation tasks, artifacts might appear, or the tissue of interest may not be segmented correctly. To address this, the framework can be used to introduce uncertainties and gain information on what is needed for an artifact to appear. Similarly, uncertainty can be introduced on a tissue of interest and see when it disappears. Furthermore, a user is able to introduce uncertainty on segmentation parameters themselves and thus express the uncertainty in the parameters, which are often hard to fine-tune.

5.4.2 Task 2.1 - Segmentation Uncertainty

The image used for this task is the slice (file "1-49.dcm") of series number 8 from the third dataset of [ZLL+21]. As a first step, we build the pipeline with only the first two modules, leaving the closing operation out for the moment. After loading the slice that includes the lesions that shall be segmented, the user manually tries to infer a good threshold intensity value by hovering over the visualization and processing without uncertainty, and decides that 90 may be a good value to split the lesions from the rest. Nevertheless, the user is uncertain and therefore introduces a PU for the lower threshold parameter, and values it with ($\mu_0 = 0$, $s_\mu = 0$, $s_n = 20$, $t = uniform$).

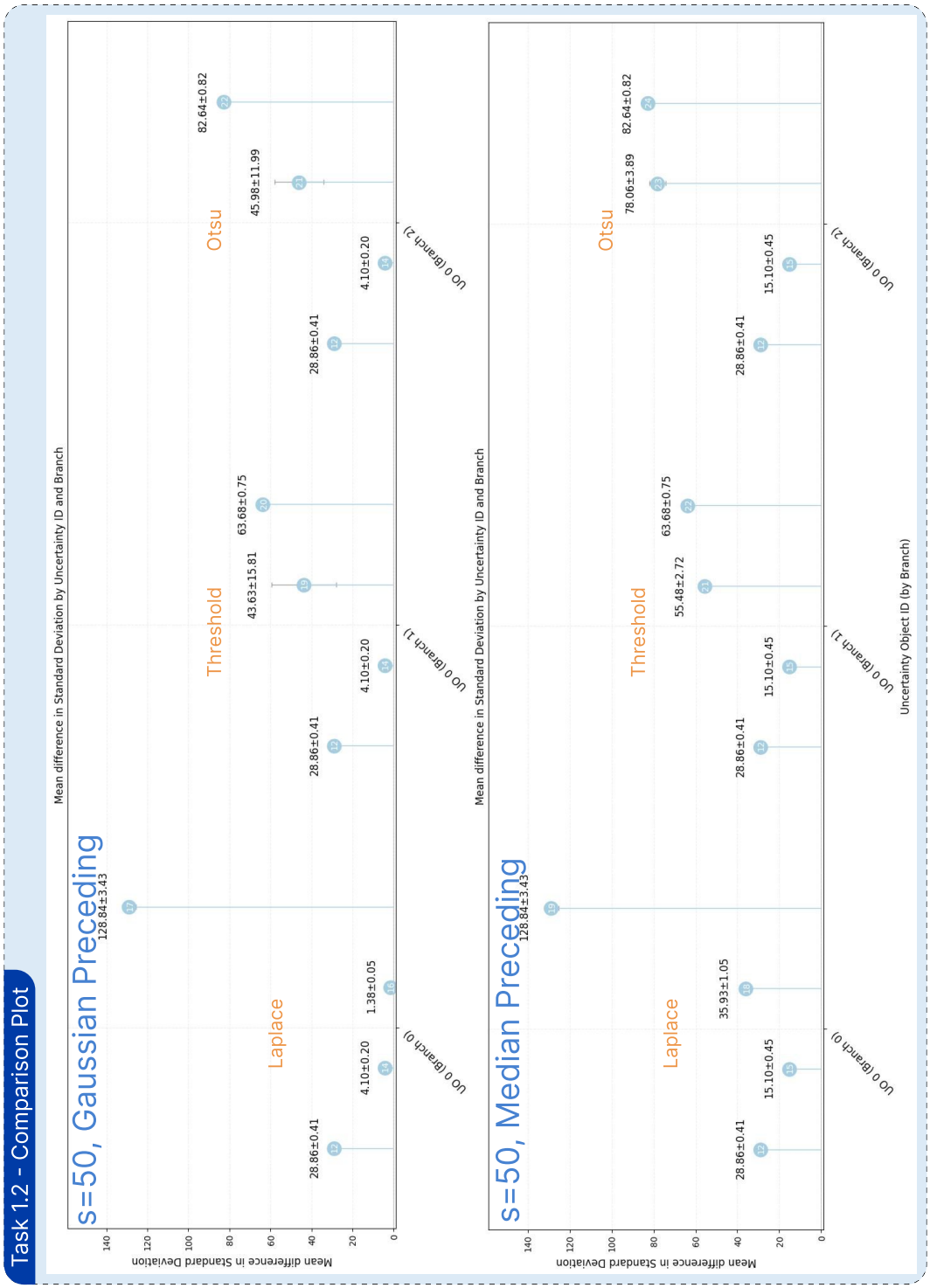


Figure 5.7: The comparison plots for Task 1.2 for an upstream connected *Gaussian Filter* and *Median Filter* in case of $s_n = 50$. In Task 1.2, the behaviors of the uncertainty-increasing modules from Task 1.1 are tested with a preceding blur module. For each module tested, there is a plot group of 4 points. The first point in the group represents the average standard deviation for the acquisition module, with the visual uncertainty included. This measure is the same in each group, as you can see, and serves as an error check for the setup. The second data point quantifies the average standard deviation when the blur module has been connected. Data point 3 shows the average standard deviation measure with the respective increasing-uncertainty module connected. And the last data point in the group measures the average standard deviation without the blur module, which corresponds to the measures from Task 1.1.

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. *Threshold Segmentation*
  3. Optional *Gaussian Filter* or *Median Filter*
  4. *Morphology Filter* Erosion
  5. *Morphology Filter* Dilation
- Uncertainties
  1. Parameter Uncertainty for the lower threshold parameter of the *Threshold Segmentation* module.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 20$, $t = uniform$
- Sample size $n = 500$

The user computes the node impact, investigates the case, and compares the pipeline results with and without uncertainty, which are shown in Figure 5.9. The first two rows show first, the result of the acquisition (which are equal), then the result of the *Threshold Segmentation* module first without uncertainty and then with uncertainty for the lower threshold parameter ($\mu_0 = 0$, $s_\mu = 0$, $s_n = 20$, $t = uniform$). The comparison provides the user with an impression of which pixels are less likely to be in it, because some pixels, especially the noisy regions, are less intense in the result with uncertainty. Meanwhile, some are slightly present in the result with uncertainty, which are invisible without uncertainty.

To get rid of the noise and small pixel artifacts, the user could try a blur like the *Gaussian Filter* or *Median Filter*, but instead decides on a closing morphology operation. The results are computed again, once without and once with uncertainty. Those without uncertainty are presented in Figure 5.10 and with uncertainty in Figure 5.11. Both

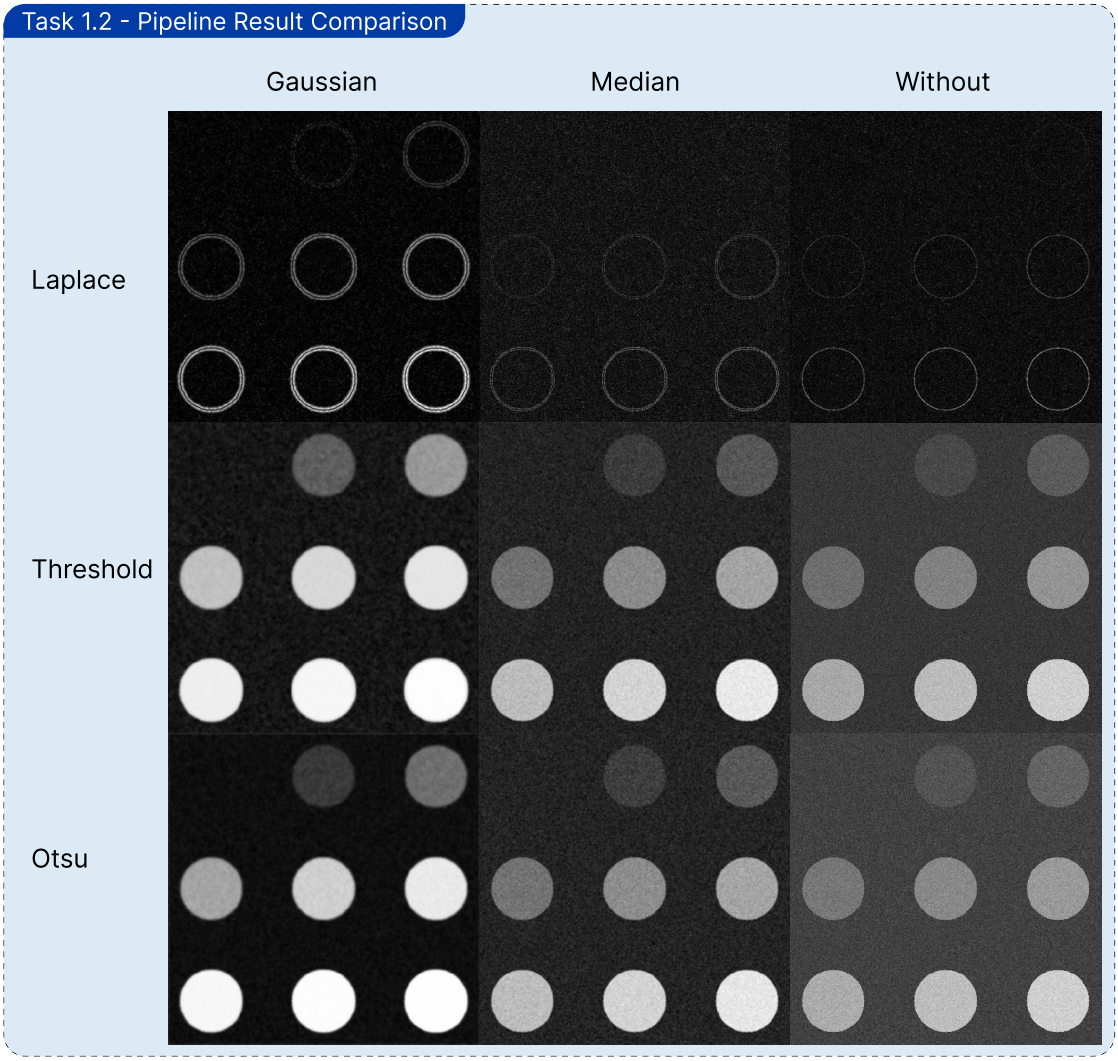


Figure 5.8: The corresponding pipeline results for Task 1.2. It shows that the *Gaussian Filter* increases the contrast between foreground and background the most. The upstream connected Median also tends to yield better results, except for the *Laplace Filter*, where it is not that definitive. Note that each image in this figure has its own optimized LUT, because the contrast can thus be judged more easily.

figures show the result of the whole pipeline step by step in row-major order (left to right across each row, then top to bottom). Ordered, we have the results after the acquisition, threshold, erosion, and dilation. The comparison of the Figures shows that the results with uncertainty yield much better results, because all lesions are visible after the closing operation, as opposed to the results without uncertainty.

Due to experience and the appearance of the dataset, the user suspects the dataset to be affected by noise. As in the previous tasks, the user changes the pipeline and tries a blur between the segmentation and closing operations. Thus, the user tries once with a *Gaussian Filter* and a *Median Filter*, and the results can be seen in Figure 5.12. The first row shows the results with uncertainty, and the second without. The first column shows the results for the *Gaussian Filter* and the second column for the *Median Filter*. Let's first look at the first row (without uncertainty). We see that the *Gaussian Filter* preserves all ten lesions, while removing the noise to a considerable degree. Meanwhile, the *Median Filter* eliminates the noise and other artifacts even better, but it loses one small lesion. In the second row (with uncertainty), we see that in general, the noise increased slightly due to the added uncertainty of the threshold. The *Gaussian Filter* even worsened compared to the result without uncertainty. Comparing the *Median Filter* to its non-uncertainty counterpart, one can see that the *Median Filter* only gained a little bit of noise while preserving all lesions.

5.4.3 Task 2.2 - Lesion Uncertainty

The user is now curious about how much uncertainty is needed for a lesion to appear or disappear, which is important to prevent unnecessary treatment or, on the other hand, potentially miss a pathology and thus in both cases worsen a patient's diagnosis and treatment. The analysis is continued based on the findings of the preceding tasks. Thus, a *Median Filter* is added between the *Threshold Segmentation* and the *Morphology Filter* Erosion. Furthermore, the values for the segmentation module are copied from Task 2.1.

Now the user places a circular VU or a lesion and a non-lesion tissue and tries to optimize the parameters for both VUs, so that the region of the lesion that shall disappear approximately matches the uncertainty of other non-lesion tissue, and that the non-lesion tissue now matches the uncertainty of the other present lesions. To accomplish this, the user switches back and forth in the view modes between the mean, standard deviation, and overlay of the standard deviation until satisfaction. The resulting uncertainty parameters of the VUs may provide information that may be used to estimate probabilities for missed lesions or wrongly segmented lesions.

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. *Threshold Segmentation*
  3. *Median Filter*

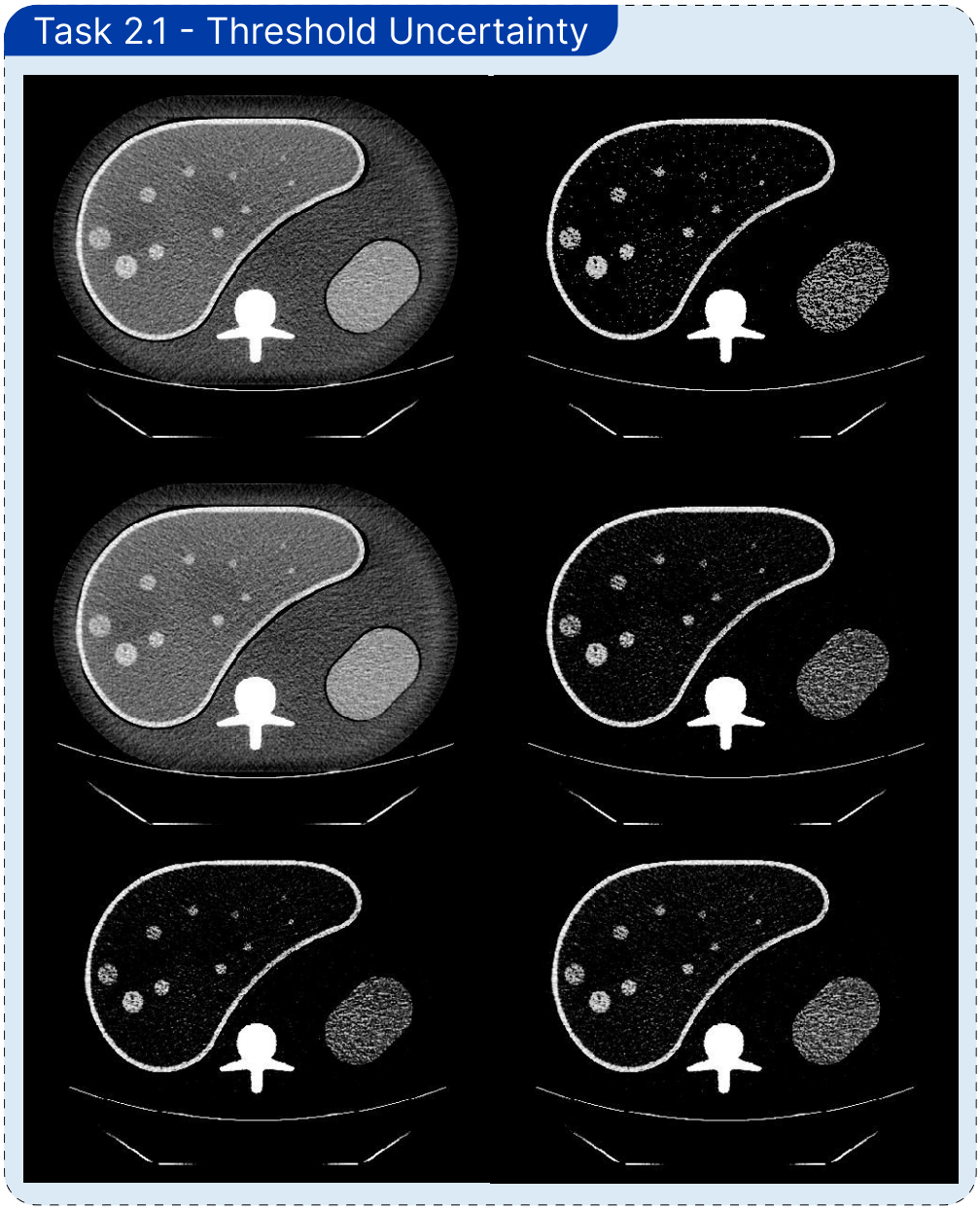


Figure 5.9: A part of the node impact analysis for Task 2.1. The slice from dataset [ZLL+21] visualizes a phantom of the liver that has 10 inserted spherical lesions of varying size and radiodensity. The first row shows the segmentation with no PU and just the manually determined threshold. The second row shows the same, but with the PU. And the last row shows the lower (left) and upper (right) confidence intervals for the segmentation with the PU.

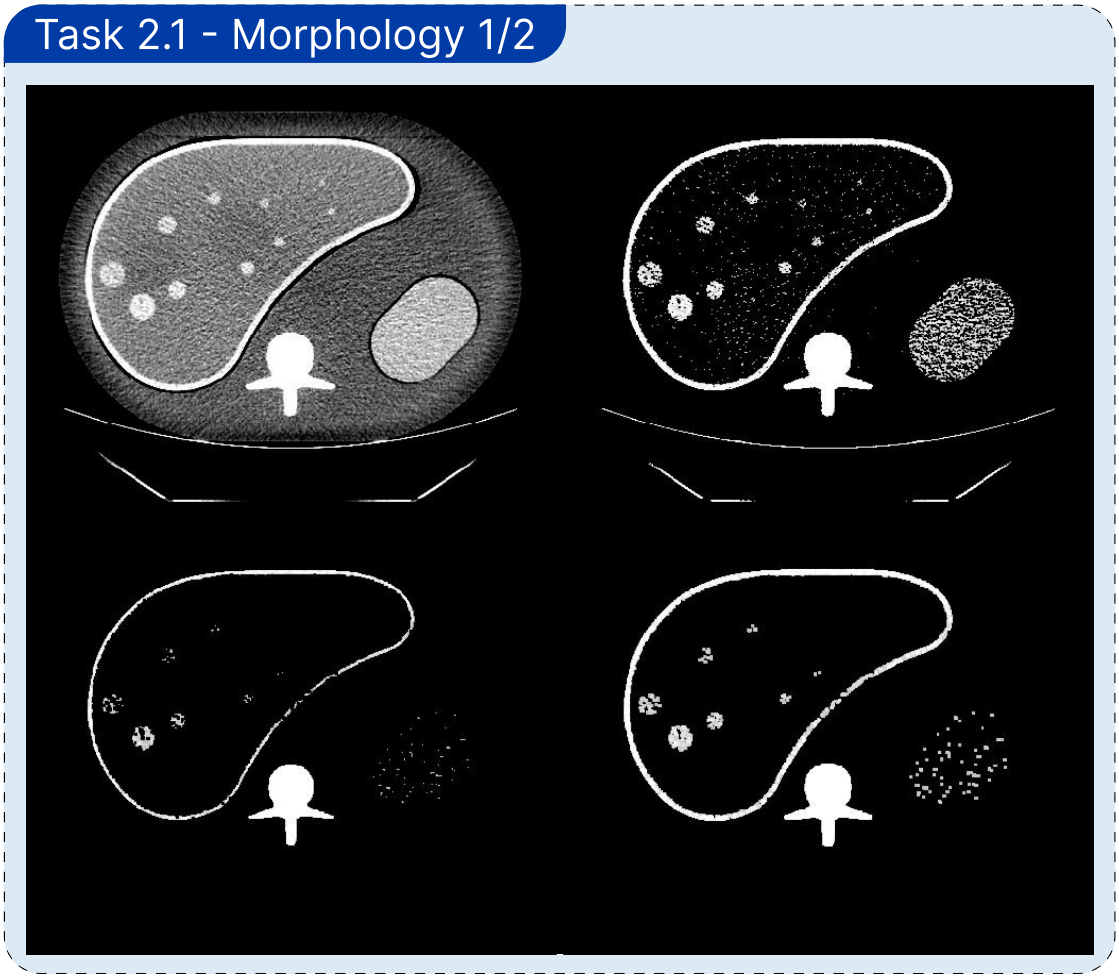


Figure 5.10: Pipeline results for the segmentation Task 2.1 without uncertainty. The order is row-major (left to right across each row, then top to bottom). First, we have the results after acquisition, second, after the threshold, followed by erosion, and then dilation.

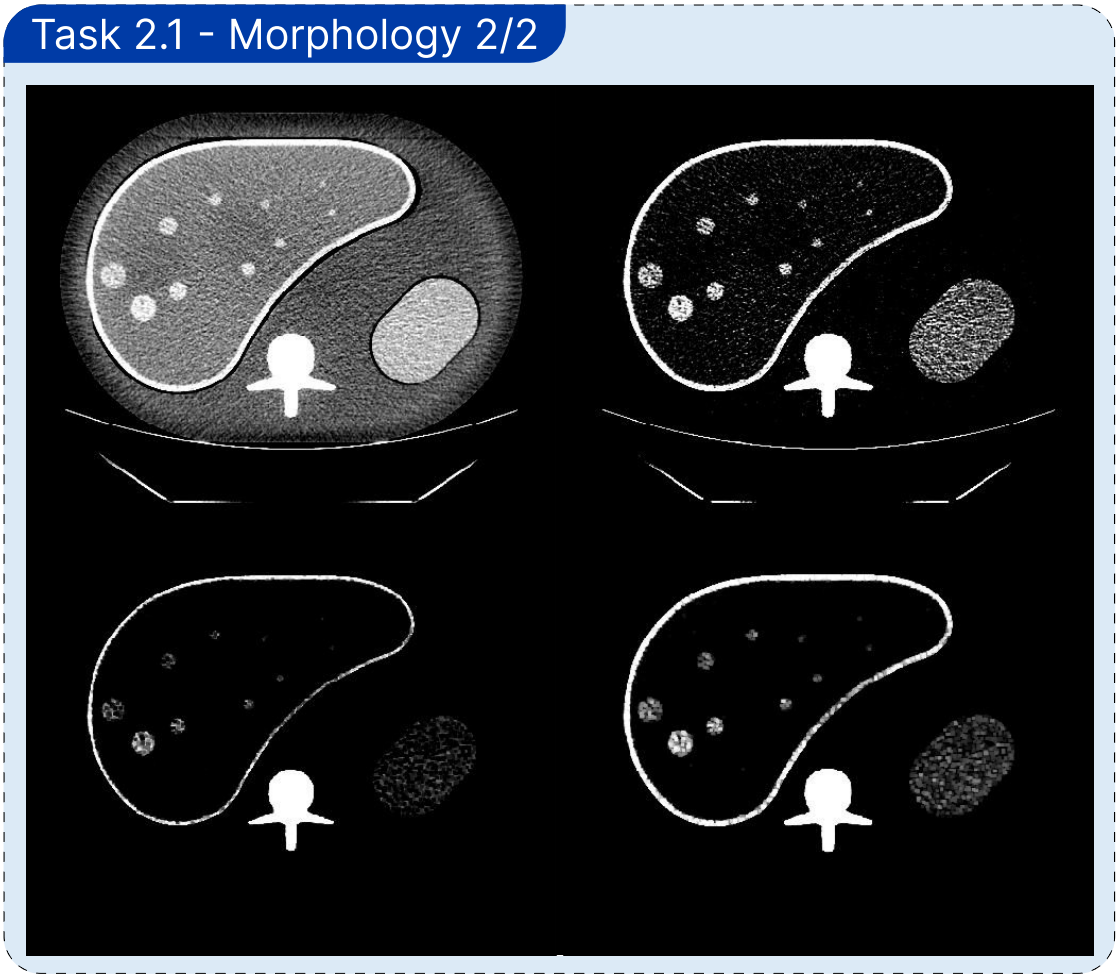


Figure 5.11: Pipeline results for the segmentation Task 2.1 with the PU on the lower threshold of the segmentation module. The order is row-major. In order, we have the results after the acquisition, the threshold, erosion, and dilation.

4. *Morphology Filter* Erosion
5. *Morphology Filter* Dilation

- Uncertainties
  1. Parameter Uncertainty for the lower threshold parameter of the *Threshold Segmentation* module.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 20$, $t = uniform$
  2. Circular Uncertainty over non-lesion tissue.
     $\mu_0 = 42$, $s_\mu = 10$, $s_n = 8$, $t = uniform$
  3. Circular Uncertainty over lesion tissue.
     $\mu_0 = -50$, $s_\mu = 5$, $s_n = 8$, $t = uniform$

Figure 5.12: Results of Task 2.1 with an interposed blur. The results without uncertainty are shown in the first row. First for the *Gaussian Filter*, then for the *Median Filter*. Analogously, the second row presents the results with uncertainty.

- Sample size $n = 500$

Figure 5.13 shows the results for this task. Row by row, there are first the parameters of the uncertainty models of the circular VUs. The darker blue circle covers a lesion, while the light blue circle covers non-lesion tissue. The second row then presents the mean results after the segmentation and dilation operation, and the last row shows the respective standard deviation overlay. Looking closely at the four images at the bottom, one can see that the manipulated areas fall roughly into the same pattern as the other non-lesion, respectively, lesion tissue regions. By adjusting the parameters, the user achieved look-alike lesions and non-lesions, thereby gaining insight into the amount of uncertainty required to make a lesion appear or disappear. As mentioned, a user might be able to further use those distribution values and combine them with prior knowledge on the acquisition uncertainty and compute probabilities for the two events.

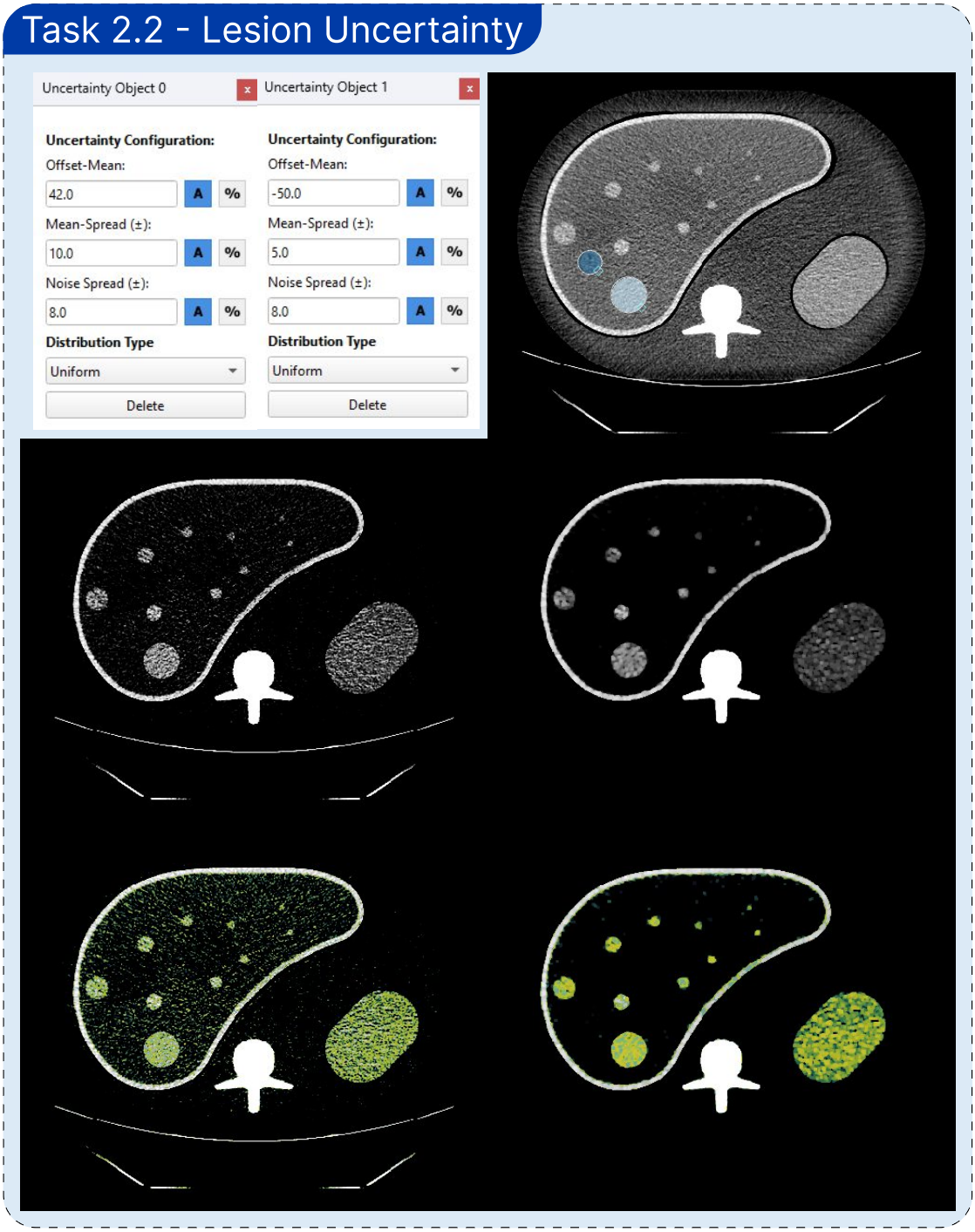


Figure 5.13: The results for Task 2.2. Left top are the configurations for the circular VUs that are visible in the top right image. Row two shows the mean after the segmentation and dilation module. Row three shows the results of the same module but now with the uncertainty (standard deviation) overlay.

5.5 Usage Scenario 3

5.5.1 Scenario Definition

Real-world applications and pipelines are often complex and depend on a lot of parameters. A user can use the framework and eventually implement missing modules or altered uncertainty models and analyze how uncertainties behave in standard real-world pipelines. Furthermore, the provenance tree with different branches of real-world pipelines can be utilized to alter parameters, compare different node impacts, and thus gain insights that may improve further real-world pipelines and increase awareness of how uncertainty impacts daily used pipelines. Being limited to the implemented modules, it is not possible to show high-performing visualization pipelines. Nevertheless, we present the idea of how such an analysis could look with the current state of implementation in Task 3.1.

Isolating changes in uncertainty parameters or module parameters and analyzing their effects can be useful. However, as discussed in Section 2.3.3 on parameter sensitivity analysis, it is important to exploit the whole parameter space with global sensitivity analysis methods and thus tweak multiple parameters. While the framework is by no means designed to provide a complete global sensitivity analysis, it is possible to analyze and compare numerous changes and thus get a glimpse of insight into the complex interfering behavior of the parameters. Similar to previous usage scenarios, users can test their specific assumptions and use the information to improve medical visualization techniques. A visualization pipeline is full of dependencies as data is processed by modules sequentially. This is relevant because the metrics computed in our approach always rely on the baseline pipeline, including all uncertainties. The analysis of multiple parameter changes is nevertheless possible, but one has to be aware of dependencies and be cautious when drawing early conclusions. In other words, it is important that users are aware that the results and metrics available in the framework may only be valid in certain cases. In Task 3.2, we therefore present special cases we discovered with the help of the framework.

5.5.2 Task 3.1 - Parameter Analysis

One significant advantage of the framework is that it allows for step-by-step modification of the pipeline, enabling the comparison of all node impacts simultaneously. This opens the door for the analysis of changes both individually and collectively, as a series of steps. A user might be interested in how the sample size n, in combination with different distribution types, influences the pipeline result metrics. Thus, first, a simple pipeline with only three VUs on a constant image is built. Each uncertainty has a distinct distribution type, but the same uncertainty parameter values. Then the sample size n is changed five times to n of 10, 100, 1000, 10000, and 100000 for which the node impact is computed.

Those node impacts are compared in average bias and average uncertainty, which are shown in Figure 5.14. The light blue (UO 0) data points represent the uniformly distributed uncertainty, the dark blue (UO 1) the normal, and the green (UO 2) the

beta distribution. The upper plot shows the grouped average standard deviation, while the lower plot shows the average bias of the difference. For each uncertainty group, the data points are ordered from low to high sample size. One can see that the average uncertainty does not change significantly with the sample sizes; it seems that they rather converge. This is expected as the variance of the sample pixels does not change with a higher sample size, but converges to the true variance. However, the average bias does change with higher n and converges for all uncertainties to the offset, which is 0 in this example. Again, this is expected because the noise of each sample interferes and, with higher sample sizes, it gets canceled out.

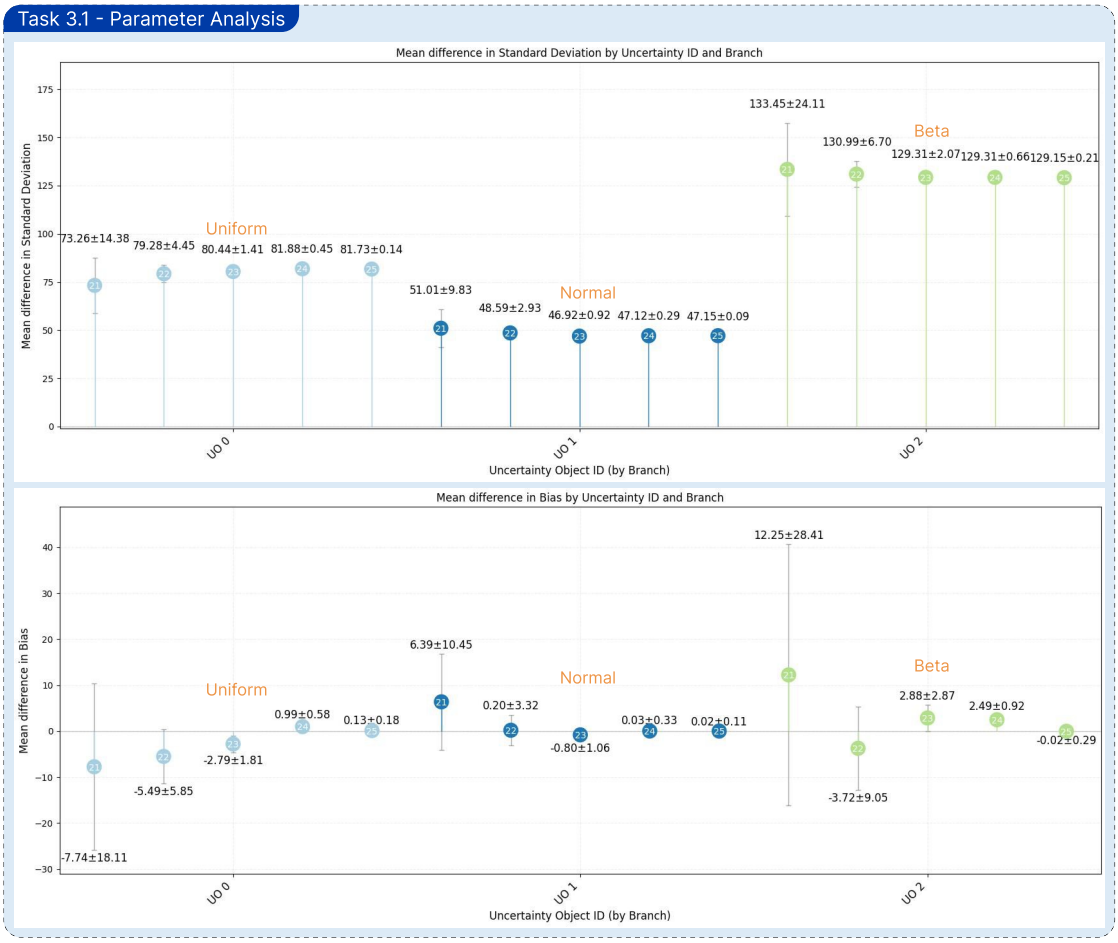


Figure 5.14: The results for Task 3.1 - Parameter Analysis. U0 0 - Uniform distribution, U0 1 - Normal distribution, U0 2 - Beta distribution. Besides that, all uncertainties have the same parameters of $\mu_0 = 0$, $s_\mu = 100$, $s_n = 100$. That setup was executed with sample sizes of $10, 100, 1000, 10000, 100000$. The upper plot shows the average standard deviations across the sample sizes grouped by uncertainties. And the lower plot analogously shows the average bias.

5.5.3 Task 3.2 - Uncertainty Metrics Analysis

One interesting interaction we already came across but skipped for the moment is in the example of Task 2.2. There, we had an uncertainty on the threshold parameter in addition to circular uncertainties to produce a fake lesion and eliminate a real lesion. Look at Figure 5.15, the darker blue uncertainty ($\mu_0 = -50$, $s_\mu = 5$, $s_n = 8$, $t = uniform$) removes a lesion and therefore lowers the intensity of the region. That turns the segmentation of the affected region to be less ambiguous, because the uncertainty drops the intensity below the threshold and, on average, reduces the variance. This is why the average variance change of the uncertainty is negative, because it lowers the uncertainty compared to when it would not be present. Additionally, one could notice that the average variance change for the VUs is 1775.49 and -1119.38, while the value for the threshold uncertainty is only 70.11. The reason is that the metrics for VUs are always calculated for the area they are affecting, thus the area of impact is much smaller for the VUs. This case aims to point out that context matters a lot when analyzing uncertainties, because in other cases, an uncertainty with such parameters as in Task 2.2 would by no means lower the average variance.

While collecting results, we encountered a mathematical theorem that we were previously unaware of. Namely, the Jensen Inequality [Mer23], although it was discovered centuries ago, we still present its occurrence as a result, because it has some implications for the result analysis of our framework. In its probabilistic form, with a random variable X and a concave function φ, the theorem summarized states

$$\varphi(\mathbb{E}[X]) \geq \mathbb{E}[\varphi(X)]. \tag{5.1}$$

And translated to our case, we have our variance difference across all pipeline stages σ^2_{diff}. Thus, we take the last by selecting layer $M-1$ and take the mean, respectively.

$$\sqrt{mean(\sigma^2_{diff}[M-1])} \geq mean(\sqrt{\sigma^2_{diff}[M-1]}). \tag{5.2}$$

In consequence, this means that in the main plot of the node comparison, in which the average variance and average standard deviation are shown, the standard deviation does not have to be equal to the square root of the average variance. This is because the square root function is concave and therefore distorts the standard deviation. The effect, however, increases a lot in cases when the distribution of the variance itself is wide. In general, the decision on which metric to use for which analysis depends on the context. While the average standard deviation still has its use, because it has the same units as the data, the variance's units are squared. Nevertheless, the variance is quite relevant because it is additive for linear cases and thus can be used to accumulate uncertainties. To show all this and more, we investigate different setups of pipelines with linear and non-linear modules, place multiple uncertainties, and see if the variance adds up to the total variance. Furthermore, we check if the Jensen Inequality theorem [Mer23] applies by testing if the standard deviation is the square root of the respective variance.

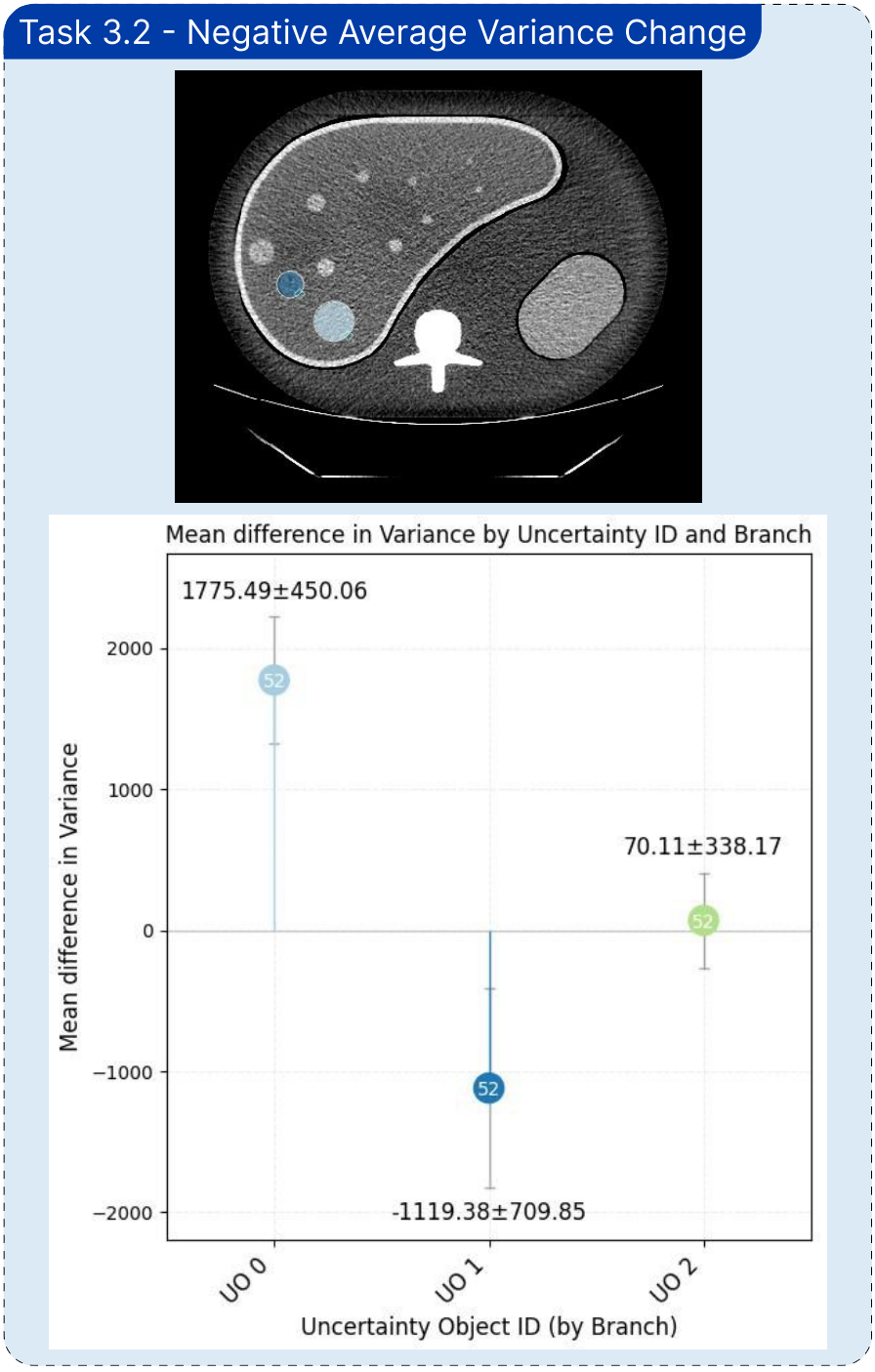


Figure 5.15: Results for Task 3.2 taken from Task 2.2. The image on top shows the placement of the VUs. The green uncertainty is the uncertainty introduced for the lower threshold value. The darker blue average uncertainty change shows that the value can also get negative, because it lowers the uncertainty of the segmentation.

Figure 5.16 shows the corresponding provenance tree, which shows the computed node impacts for the 5 testing configurations (marked with a small purple circle). The first part of the test consists of 5 different pipeline setups that all include the same VUs and vary in the modules used. The pipeline always comprises four stages, and for the first three stages, there is one rectangular VU introduced that spans across the whole image. As an image source, the *Circle Generator* is used with a high radius so that we get a homogeneous image.

Pipeline Setup

- Module Order
  1. *Circle Generator*
  2. Stage 1 Module
  3. Stage 2 Module
  4. Stage 3 Module
- Uncertainties
  1. Rectangular Uncertainty
     $\mu_0 = 0,\ s_\mu = 0,\ s_n = 60,\ t = uniform$
  2. Rectangular Uncertainty
     $\mu_0 = 0,\ s_\mu = 0,\ s_n = 40,\ t = uniform$
  3. Circular Uncertainty.
     $\mu_0 = 0,\ s_\mu = 0,\ s_n = 20,\ t = uniform$
- Sample size $n = 10000$

The first configuration uses *Gaussian Filters* for the stages 1–3, but all with $\sigma = 0$, which has the effect that they do not influence the data and therefore are "blank modules". The results are shown in Figure 5.17. On the left side, the average variances of the three rectangular VUs are shown, and on the right side, the average standard deviations. In the top right corner, the total average variance and average standard deviation of the result are noted. The summation of the three variances is 1866.59, which matches the total average variance. Since standard deviations are not additive, they should not add up, as their summation of 69.2 proves. But they provide a measure for the magnitude of the uncertainty. Remember that the uncertainties use the magnitudes of $s_n = 60$, $s_n = 40$, and $s_n = 20$. Those are approximately reflected in the scaling of the resulting average standard deviations by taking the value of the uncertainty with the lowest standard deviation 11.54 (green uncertainty in the right plot) and scaling it accordingly to the uncertainty magnitudes as follows $11.54{*}2 = 23.08$, and $11.54{*}3 = 34.62$. Then one can see that the scaling of the average standard deviations matches the scaling of the uncertainty magnitudes. Lastly, the Jensen Inequality theorem is tested by taking the square root of the variances. The results align very well, which is expected because the distributions

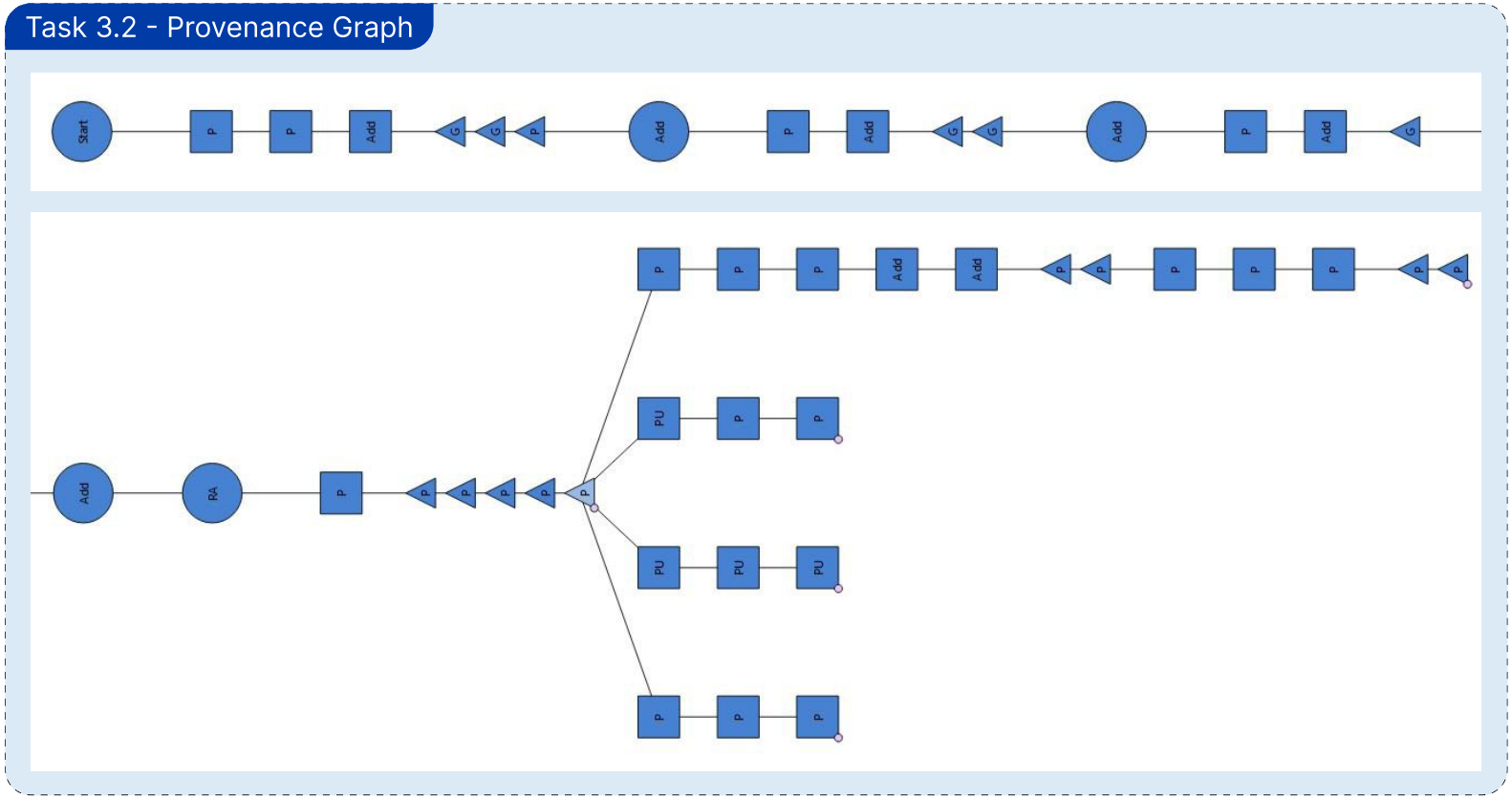


Figure 5.16: The provenance tree for Task 3.2, which includes the computed node impacts for the 5 testing configurations (marked purple). Due to visualization purposes, the tree is split in half, where the upper is the first half and the lower is the second half.

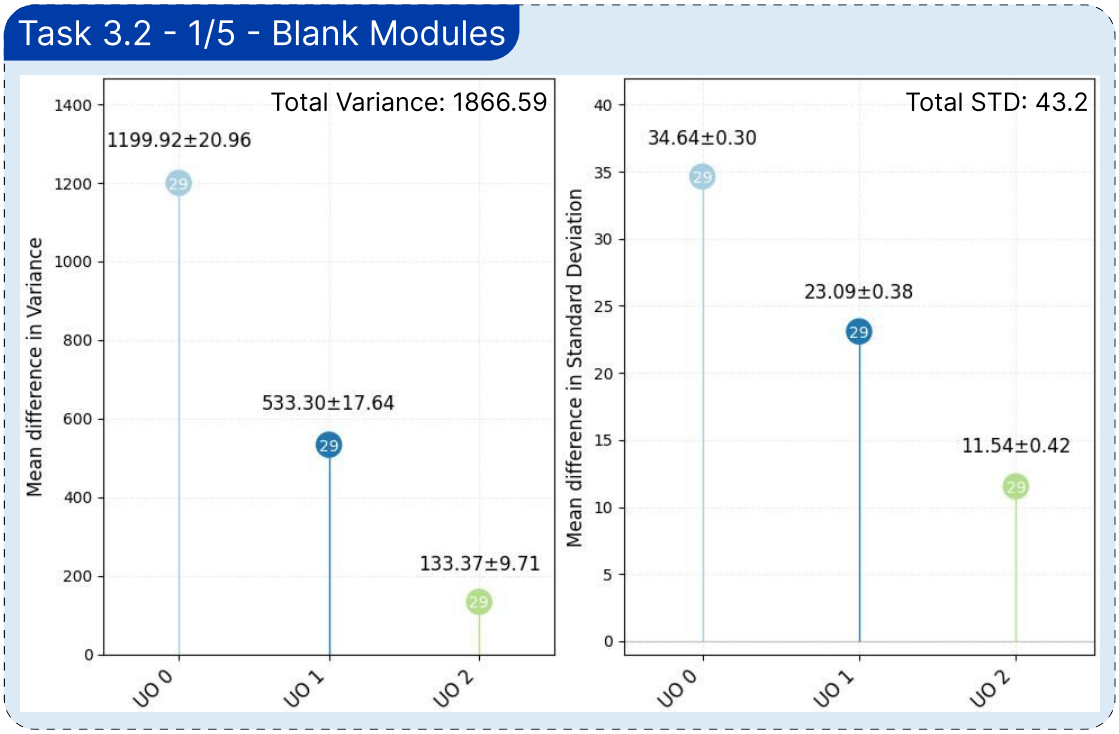


Figure 5.17: The results of the first configuration of Task 3.2, which includes the three uncertainties on three black modules. The left plot shows the average variance, and the right the average standard deviations for the three uncertainties. The important insights of this figure are that the average variances *add up* to the total variance, the square roots of the average variances on the left *align* with the standard deviations on the right, and that the standard deviations on the right *reflect* the scaling of the uncertainty magnitudes on the input, which are $s_n = [60, 40, 20]$.

are quite narrow and their shape is intact ($\sqrt{1199.92} = 34.64$, $\sqrt{533.3} = 23.09$, and $\sqrt{133.37} = 11.55$). This is verified by taking the square roots of the variances in the left plot, as shown, and comparing them to the standard deviations on the right. The pattern of how the criteria (variance additivity, standard deviation magnitude scaling, and Jensen Inequality Gap) are tested is similar fashion in the following configurations.

In the second configuration, we change the σ values of stages 1–3 to $\sigma = 2$ and test it similarly to the previous configuration. Since *Gaussian Filters* are linear, the measures should still align as in the previous example. Figure 5.18 shows the results of the second part. The summation of the variances aligns with the result of 16.77. While the standard deviations don't add up, they also do not reflect the scale of magnitude anymore ($1.65 * 2 = 3.3$, and $1.65 * 3 = 4.95$). But since the pipeline only consists of linear operators that do not distort the distributions, the square roots of the variances match well ($\sqrt{8.45} = 2.9$, $\sqrt{5.57} = 2.36$, and $\sqrt{2.75} = 1.66$).

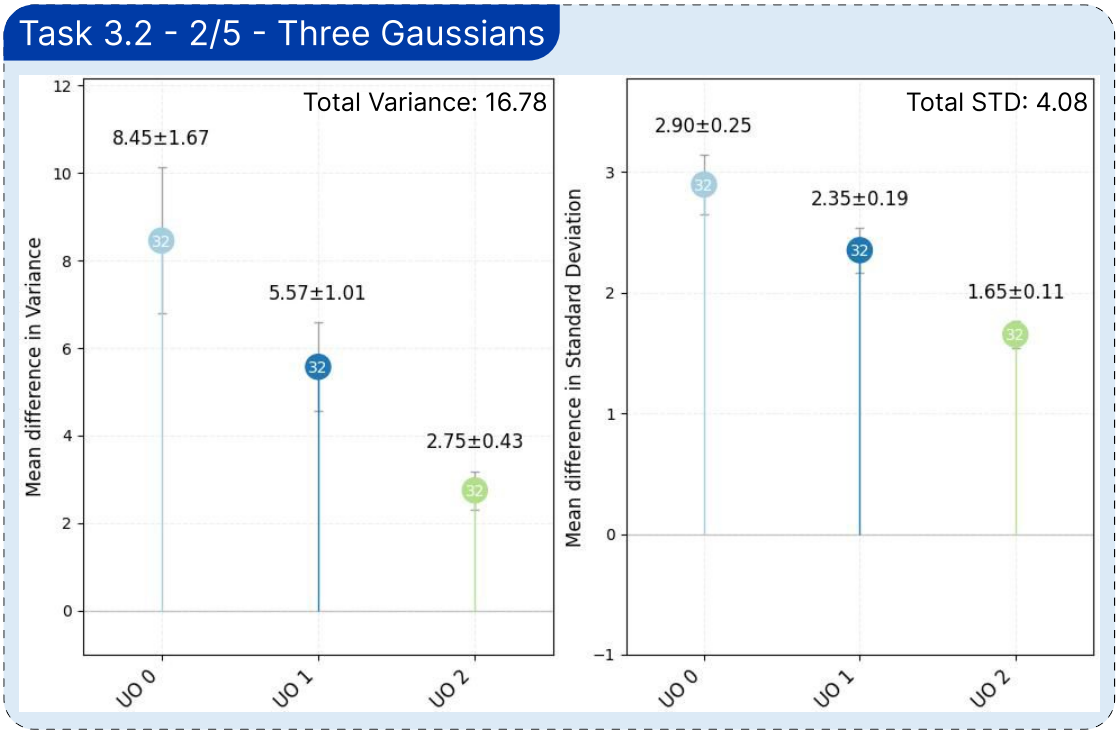


Figure 5.18: The results of the second configuration of Task 3.2, which includes the three uncertainties in combinations with three *Gaussian Filters*. The left plot shows the average variance, and the right the average standard deviations for the three uncertainties. The important insights of this figure are that the average variances *add up* to the total variance, the square roots of the average variances on the left *align* with the standard deviations on the right, and that the standard deviations on the right *do not reflect* the scaling of the uncertainty magnitudes on the input, which are $s_n = [60, 40, 20]$.

The third configuration now changes the modules to *Median Filters*, which are non-linear and therefore should break the additive behavior of the variances. The results are visible in Figure 5.19. The summation of the variances does not align anymore with a value of 142.6. Surprisingly, the scaling of the standard deviations aligns quite well ($3.06 * 2 = 6.12$, and $3.06 * 3 = 9.18$) with the scaling of the uncertainty magnitudes. Testing the gap of the Jensen Inequality results in $\sqrt{94.55} = 9.72$, $\sqrt{38.44} = 6.2$, and $\sqrt{9.5} = 3.08$, which shows that the gap is still minor due to the low spread in the underlying distribution of the average variance image.

The fourth configuration includes a one *Laplace Filter* and two *Gaussian Filters*. The configuration aims to show that the fact that the standard deviations scale according to the uncertainty magnitudes relies heavily on the modules' operation independently of their linearity. Figure 5.20 shows the corresponding results. Summing up the variances works out again (8.68). Meanwhile, the scaling of the standard deviation is not given anymore, which is visible directly in the plot without any calculations. Furthermore, the

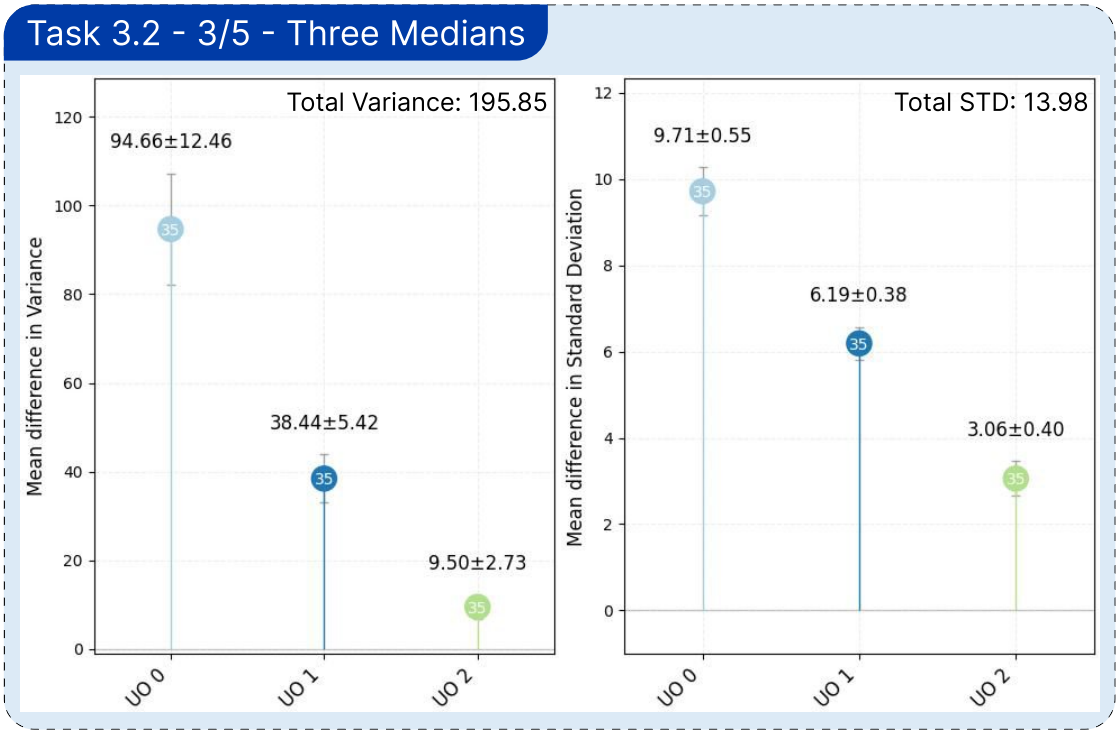


Figure 5.19: The results of the third configuration of Task 3.2, which includes the three uncertainties and three *Median Filters*. The left plot shows the average variance, and the right the average standard deviations for the three uncertainties. The important insights of this figure are that the average variances *do not add up* to the total variance, the square roots of the average variances on the left *align* with the standard deviations on the right, and that the standard deviations on the right *reflect* the scaling of the uncertainty magnitudes on the input, which are $s_n = [60, 40, 20]$.

Jensen Gaps shows to be minor with $\sqrt{0.36} = 0.6$, $\sqrt{5.57} = 2.36$, and $\sqrt{2.75} = 1.66$

The fifth configuration includes three *Gaussian Filters* and an PU for the sigma of the first and second filter (both with $\mu_0 = 0$, $s_\mu = 0$, $s_n = 1$, $t = uniform$). Accumulating the variances from Figure 5.21 results in 104.07, which does not align with the total variance. It seems that the uncertainty breaks the linear behavior of the *Gaussian Filter*, which is an interesting insight. Meanwhile, the Jensen Gap is again minor with $\sqrt{44.38} = 6.66$, $\sqrt{24.53} = 4.95$, $\sqrt{10.78} = 3.27$, $\sqrt{10.81} = 3.29$, and $\sqrt{13.57} = 3.68$.

The final part aims to demonstrate a more significant Jensen Inequality gap by creating a skewed distribution. An easy way to achieve that is with a segmentation step that includes a PU. The pipeline setup is as follows:

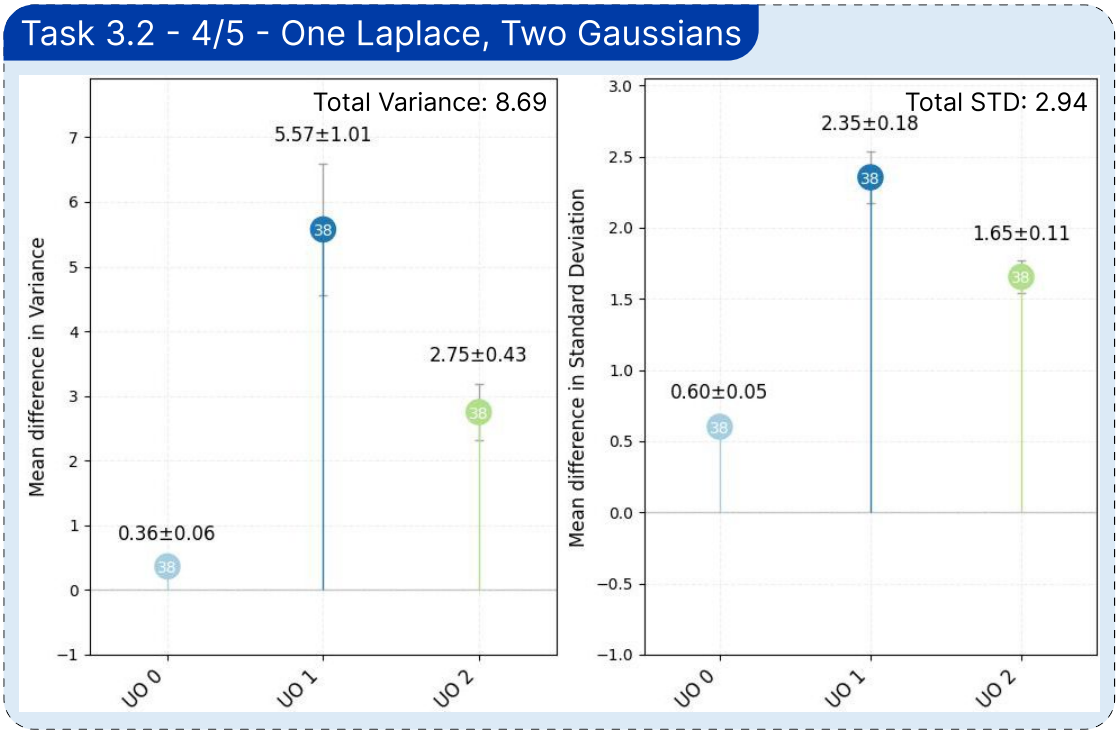


Figure 5.20: The results of the fourth configuration of Task 3.2, which includes the three uncertainties in combination with a *Laplace Filter* and two *Gaussian Filters*. The left plot shows the average variance, and the right the average standard deviations for the three uncertainties. The important insights of this figure are that the average variances *add up* to the total variance, the square roots of the average variances on the left *align* with the standard deviations on the right, and that the standard deviations on the right *do not reflect* the scaling of the uncertainty magnitudes on the input, which are $s_n = [60, 40, 20]$.

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. *Gaussian Filter*
  3. *Threshold Segmentation*
  4. *Morphology Filter* Erosion
  5. *Morphology Filter* Dilation
- Uncertainties
  1. Parameter Uncertainty for sigma parameter of the *Gaussian Filter* module.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 2$, $t = uniform$
  2. Parameter Uncertainty for the upper threshold parameter of the *Threshold Segmentation* module.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 30$, $t = uniform$
- Sample size $n = 10000$

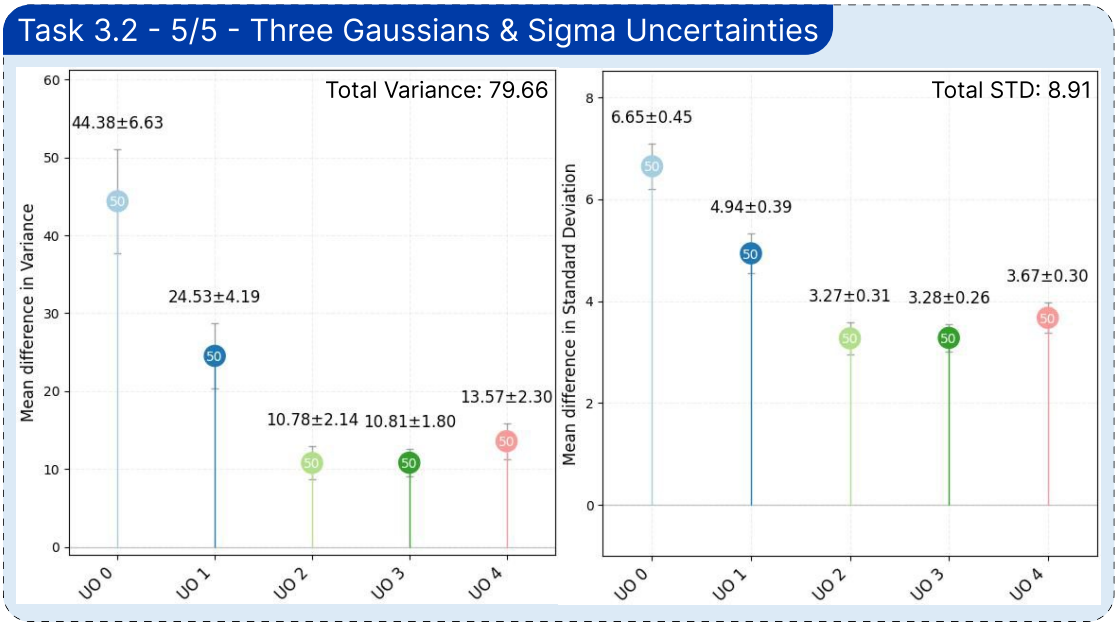


Figure 5.21: The results of the fifth configuration of Task 3.2, which includes the three uncertainties in combination with three *Gaussian Filters* where the first two have a PU on the parameter σ. The left plot shows the average variance, and the right the average standard deviations for the three uncertainties. The important insights of this figure are that the average variances *do not add up* to the total variance, the square roots of the average variances on the left *align* with the standard deviations on the right, and that the standard deviations on the right *do not reflect* the scaling of the uncertainty magnitudes on the input, which are $s_n = [60, 40, 20]$.

Figure 5.22 and Figure 5.23 show the corresponding results. The blue uncertainty is the sigma uncertainty, and the green is the uncertainty for the lower threshold. First of all, one sees that the variances do not add up to the total variance. Furthermore, the square roots of the variances do not match ($\sqrt{9932.03} = 99.66$, and $\sqrt{6728.6} = 82.03$), which represents a bigger Jensen Gap. In Figure 5.23, on the left side, the variance distribution for an uncertainty of this example is shown, and on the right, a variance distribution from Figure 5.21. Note that the y-scale from the left distribution is logarithmic. We could have shown all distributions from the previous part of this task, but they all have a normal distribution-like shape, because they do not include modules that have a cutoff effect. Whereas the left distribution is quite skewed and therefore amplifies the Jensen Gap.

All in all, we were able to find out that the proposed methodology allows the addition of variances if the modules are linear; however, uncertainties on linear modules can break this behavior. Furthermore, in some cases, the scaling of the standard deviations matches the scaling of uncertainty magnitudes. Lastly, we have seen that due to the Jensen Inequality, a gap between the square root of the average variance and the average

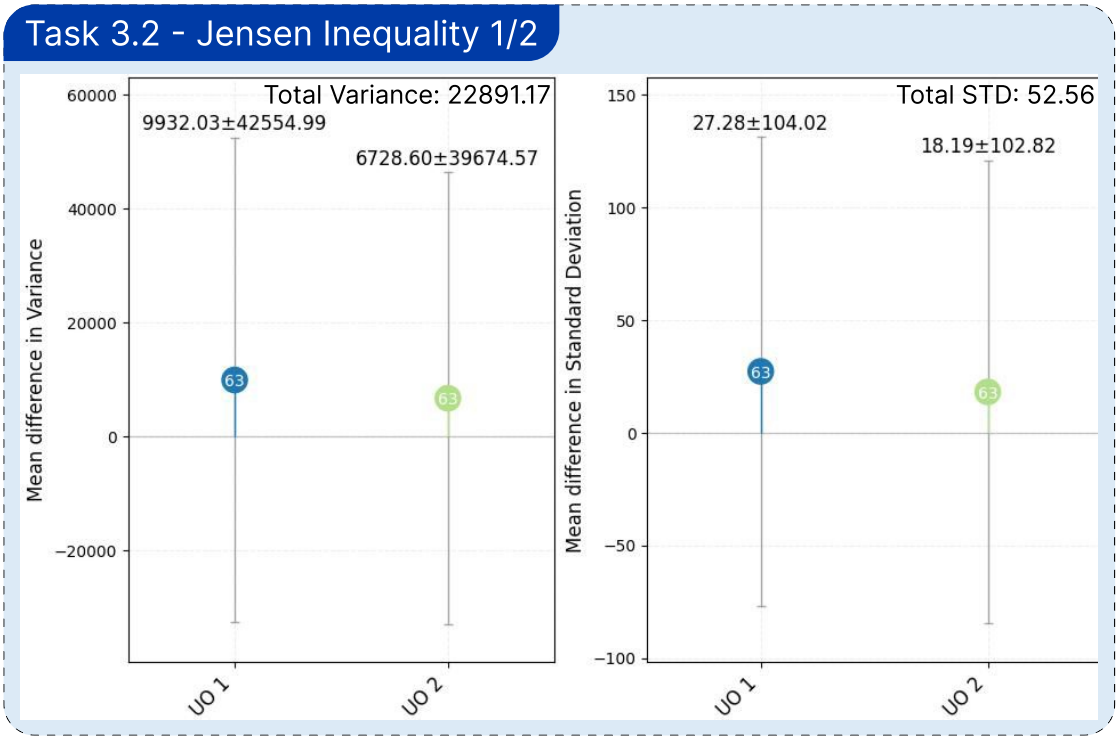


Figure 5.22: The first part of the results from the Jensen Inequality demonstration of Task 3.2. The left plot shows the average variance, and the right the average standard deviations for the two uncertainties. The important insights of this figure are that the average variances *do not add up* to the total variance, and that the square roots of the average variances on the left *do not align* with the standard deviations on the right.

standard deviation exists, which takes effect especially the more skewed the distribution is.

5.6 Performance Analysis

As the methodology is computationally intensive by design, we conducted a simple performance analysis on the most important parameters: image resolution, sample size, and uncertainty count. Theoretically, we anticipate the framework to scale linearly with each of the three. The image resolution quadratically increases the pixel count and thus the workload. The sample size determines how many times the pipeline is processed and thus also has a linear impact. As previously explained in the methodology, the pipeline is run n times (sample size) per uncertainty, plus one time, which also would be linear.

The base pipeline setup for the analysis is described in the following. From there on, we adjust according to the variable parameters. Each test setting has been run 3 times, and results have been averaged to eliminate outliers.

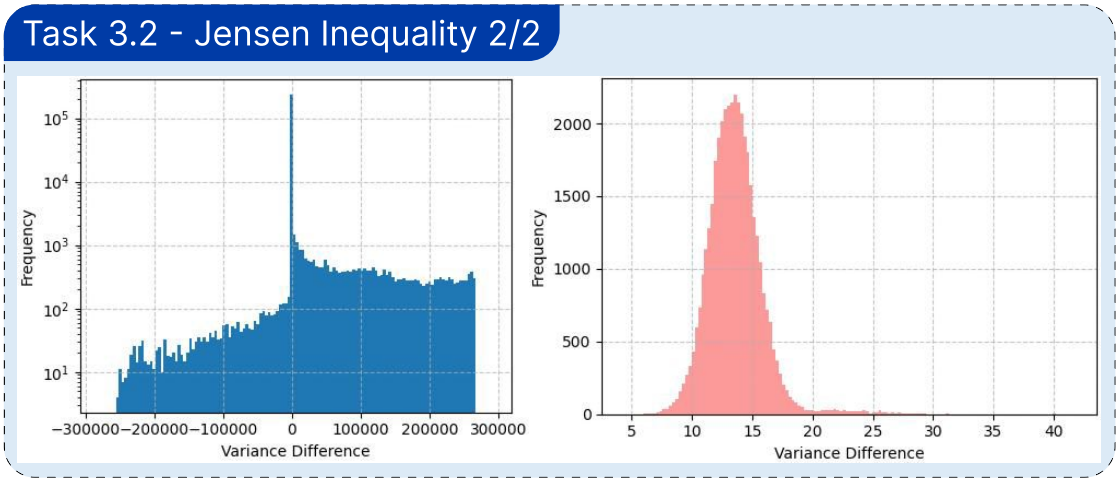


Figure 5.23: The second part of the results from the Jensen Inequality demonstration of Task 3.2. The left plot shows the variance distribution of an uncertainty from this example, while the right side shows an example from a previous configuration of Task 3.2, which kept the Jensen Inequality gap minor. Note that the left distribution has a logarithmic y-axis. One can see that the left side distribution is much more skewed and distorted than the right one. Thus, the Jensen gap appears to be much bigger for the underlying data of the left variance distribution than the right. To confirm this, compare Figure 5.22 and Figure 5.17

Pipeline Setup

- Module Order
  1. *Image Acquisition*
  2. *Gaussian Filter*
  3. *Threshold Segmentation*
  4. *Morphology Filter* Erosion
  5. *Morphology Filter* Dilation
- Uncertainties
  1. Parameter Uncertainty for the lower threshold parameter of the *Threshold Segmentation* module.
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 1$, $t = uniform$
  2. Rectangular Uncertainty
     $\mu_0 = 0$, $s_\mu = 0$, $s_n = 1$, $t = uniform$
- Sample size $n = 500$

The results are shown in Figure 5.24, in which for each sample size (top left), side

length (top right), and uncertainty plot (bottom left), the average respective benchmark time is plotted. In the bottom right, the corresponding average values are denoted for clarification. The anticipated behavior is backed up by the results shown. The process time scales linearly with the sample size (top left) and uncertainty count (bottom left). For the side length (top right), the scaling is quadratic relative to the side length, but as the pixel count is the side length squared, the process time scales linearly with the pixel count.

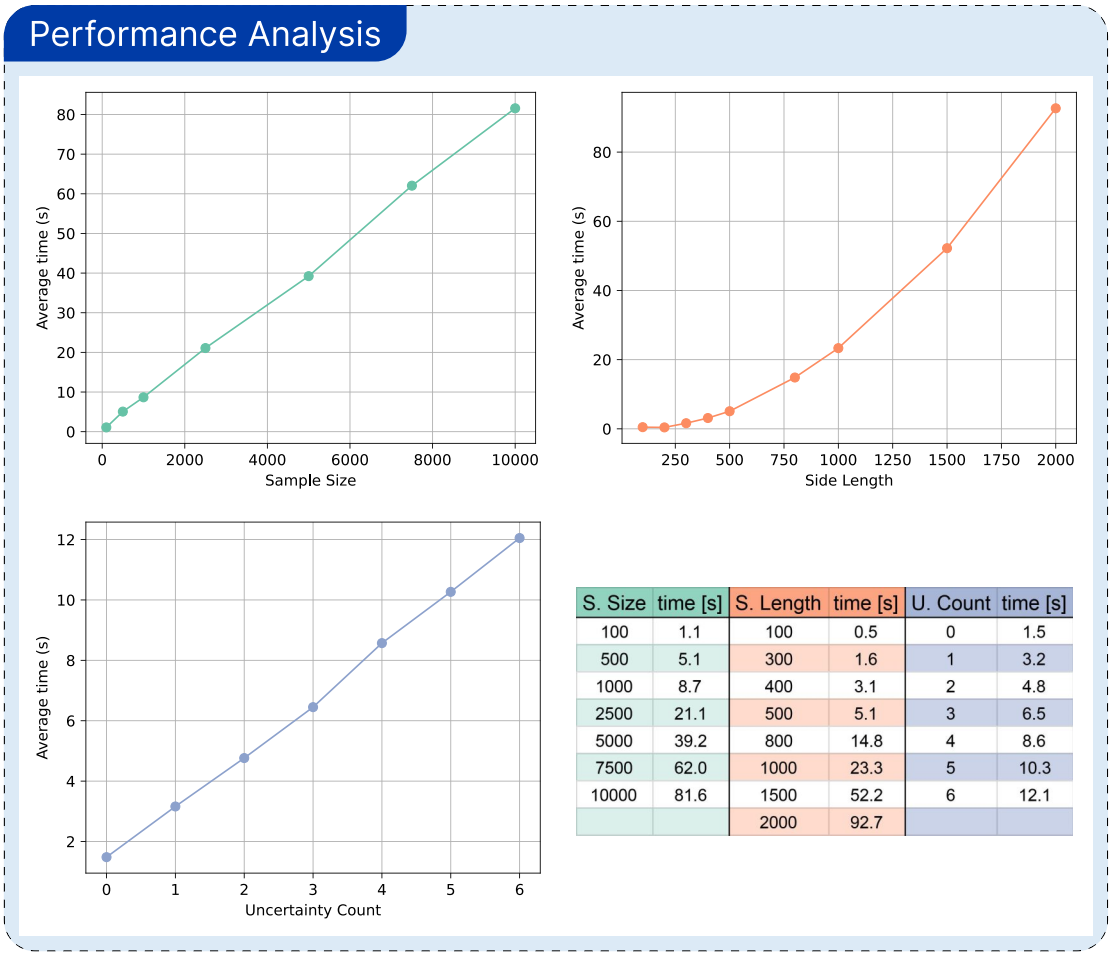
Despite the alignment with the theory, we overlook effects like module performance or pipeline performance in general. Since the pipeline is modular, the performance can vary a lot based on the pipeline setting, which especially applies to more advanced and currently not implemented modules. Thus, the results have to be taken into account with caution and reevaluated for other settings.

5.7 Discussion

Usage Scenario 1 shows a possible way for a module analysis. A user can compare different options and investigate if trends for specific modules show up, and by extending the analysis, how dependent their respective uncertainty effect is on the pipeline setup itself. In Usage Scenario 2, the concept of how a user could utilize uncertainty in combination with provenance to optimize a segmentation task is laid out. Furthermore, it shows how it may be possible to estimate the probability for the event of a fake lesion appearing or a lesion disappearing. Usage Scenario 3 includes results that allow the analysis of the behavior of the average variance and standard deviation as uncertainty measures. By investigating the results, we have seen that the variances are additive if pipelines are linear. Additionally, the Jensen Inequality gap has been shown to grow the more a distribution is distorted. Lastly, the standard deviation has been shown to reflect the scaling of magnitudes of the uncertainties in some cases. Nevertheless, a clear correlation has not been found.

In the presented Usage Scenarios and their sub-tasks, possible usages of the framework have been shown. The combination of an uncertainty propagation analysis and provenance offers a promising way to inspect certain use cases and thus gain insights to fine-tune a pipeline. While the pipelines with a higher complexity could not be shown, the proof of concept has been demonstrated with a basic toolset. The addition of the provenance graph has proved to be useful, enabling rapid development of pipelines while being able to go back and forth seamlessly. If a pipeline state seems interesting, the node impact can be computed and, therefore, compared to other states, and thus compare the respective results of uncertainty propagation.

Interestingly, the use of uncertainty in segmentation may already be useful. If a voxel consistently exceeds the threshold, it will be included in each segmentation, leading to high average intensity; if it consistently falls below, it will be excluded, resulting in low average intensity. By adding noise to the threshold (i.e., by adding perturbations) and repeating the segmentation multiple times, an average image can be generated where



S. Size	time [s]	S. Length	time [s]	U. Count	time [s]
100	1.1	100	0.5	0	1.5
500	5.1	300	1.6	1	3.2
1000	8.7	400	3.1	2	4.8
2500	21.1	500	5.1	3	6.5
5000	39.2	800	14.8	4	8.6
7500	62.0	1000	23.3	5	10.3
10000	81.6	1500	52.2	6	12.1
		2000	92.7		

Figure 5.24: Performance analysis results. *Top left:* average time versus sample size. *Top right:* average time versus image side length. *Bottom left:* average time versus uncertainty count. *Bottom right:* Detailed values of the average performance results of each plot. The time to process scales linearly in the case of the sample size and uncertainty count. For the side length, the time scales quadratic, which yields linear scaling with respect to the pixel count.

ambiguous regions (e.g., lesions) show mid-level intensities. These can then be extracted by applying a threshold to the averaged result, offering a simple yet effective strategy for segmentation refinement.

5.8 Limitations

In this section, we want to review the limitations of the framework. Most limitations are due to the fact that the framework is a proof-of-concept solution, being the first investigation of this research direction. Moreover, a lot of the limitations are a consequence of design choices at the beginning of development. Thus, we discuss those decisions retrospectively.

Dimensionality and Performance

As the framework is a proof of concept, we decided to build the framework only for two-dimensional data to avoid running into performance issues. This turned out to be a good choice, because computing a node impact for a pipeline with multiple uncertainties, a moderate sample size, and moderate resolution can quickly run out of hand.

Modules and Availability

Since the set of modules in the framework is limited to a few basic building stones, it is not possible to build realistic pipelines with state-of-the-art building blocks. This, in consequence, prevents testing the methodology on more advanced pipelines. Although it might have been feasible to add a few more complex modules, the sheer number of potential components in the medical visualization pipeline means the debate would undoubtedly have continued.

Pipeline Building

Currently, one can only build a purely sequential pipeline, i.e., one where step A runs, then B, then C, and so on. This approach does not support branching, merging, or other workflows. This, however, is a common practice in medical imaging in e.g., image registration to align datasets. The adjustment would involve redesigning the pipeline building of the framework, allowing for parallel pipeline branches and thus enabling, among others, multimodal pipelines or branching/merging operations.

Uncertainty Model

While the uncertainty model is well-suited to simulate PUs, there are limitations for VUs. The two-dimensional regions functionality can employ a low-frequency bias in the form of the mean-spread and high-frequency noise as the noise-spread. However, noise often appears structured, e.g., acquisition artifacts [PR25]. The possibility of such medium-frequency textural noise is missing in the framework. If the framework is extended, this would be the next logical step for further development to integrate an uncertainty model that has an adjustable noise frequency.

CHAPTER 6

Conclusions & Future Work

This chapter concludes the thesis by summarizing the most important aspects of the work and its takeaways. First, the methodology in its core is summarized, followed by the answers to our research questions based on our findings in the results chapter. The last section wraps up this thesis and proposes different avenues for possible future work.

6.1 Summary

To perform uncertainty propagation, we employed a basic Monte Carlo methodology. To do so, parameters or regions affected by uncertainty are sampled n times and processed by the pipeline. Across the pipeline runs, we compute the pixel-wise mean and variance and visualize them in addition to the standard deviation and confidence intervals. With the help of provenance, one can comprehend the building process of the pipeline. If a pipeline state (provenance node) seems interesting, the node impact can be computed. The node impact is the collection of statistics that are gathered when the pipeline is run once for each uncertainty, excluding that specific uncertainty, in addition to the statistics with all uncertainties included. By subtracting each excluded uncertainty result from the full uncertainty result run, we compute the following measures from the difference: the average bias, average standard deviation, and average variance. Those measures then quantify the impact of the respective uncertainty compared to when it would not be included. Thereby, our contribution lies in the development of the interactive framework as a proof of concept by combining interactive pipeline building with provenance and thus enabling uncertainty propagation comparison across distinct pipeline states.

(RQ1) *How does uncertainty propagate through the steps of the (medical) visualization pipeline?*

In the results, we presented various ways to analyze how uncertainty propagates. In Usage Scenario 1 (Section 5.3), we demonstrate a possible method to investigate modules in their

uncertainty behavior on a specific uncertainty type in a cumulative manner by testing the improvement of the current step to the next step of the analysis. In Usage Scenario 3 (Section 5.5), we chain different uncertainties together and investigate the behavior of the uncertainty measures. The findings are that the average variance (uncertainty) in linear settings is additive; thus, the average variance measures for uncertainties that affect the same area can be added together and will result in a sum close to the total average variance. One explanation for that minor margin that we became aware of is the Jensen Inequality, which grows substantially if the pipeline includes modules (e.g., segmentation modules) that skew and distort the distribution of average variance (see Usage Scenario 3 Section 5.5). Another observation made is the fact that in some cases, linear and non-linear, the average standard deviation can and cannot replicate the scaling of the magnitude of the uncertainty models themselves. In Usage Scenario 2 (Section 5.4), we show how users could utilize the framework to express their uncertainties and thus optimize their pipelines for specific scenarios by choosing modules that handle uncertainty better. Furthermore, we show how a user could possibly use uncertainty to get an estimate of the probability of a tissue being segmented incorrectly.

(RQ2) *How can we integrate provenance solutions into the (medical) visualization pipeline to track and manage uncertainty effectively?*

In the realization of the framework, we showcase how provenance can be combined with an interactive visualization pipeline to build a powerful concept to analyze uncertainty propagation. Each change to the pipeline is captured as provenance and visualized along with the other provenance data as a hierarchical tree. That tree allows a user to rapidly navigate back and forth between pipeline states and thus has similarities with an advanced undo-redo system, and thus has proven to be very useful with the mentioned on its own. In addition, a user can utilize the provenance tree to compute the node impact for nodes of interest and compare them. In the comparison, the uncertainty measures of the nodes are grouped first by the branch and then by the uncertainties present in the respective nodes. Using that concept, one can compare a flexible number of pipeline changes and analyze the impact of those based on the uncertainty measures in the comparison (see Task 3.1 Section 5.5.2). The principle of using the provenance graph to compare the uncertainty propagation analysis of multiple pipeline states has similarities to parameter sensitivity analysis. That is because the pipeline changes in the provenance tree can represent structured parameter changes of multiple nodes, whose results can be compared (see Task 3.1 Section 5.5.2). Despite that, it is important to be aware that uncertainty propagation and parameter sensitivity analysis intend to accomplish distinct goals. In uncertainty propagation, it is about discovering the impact of uncertainties on the model (pipeline), while parameter sensitivity analysis tries to unravel the principal behavior of the model by varying parameters over continuous runs. The described node impact comparison mechanism has been used for all results. While the provenance tree with its interactivity has been kept simple, the basic functionality on its own has proven to be great. Especially, navigation between pipeline states, loading, and saving made the results easily reproducible.

6.2 Future Work

The framework only serves as a proof of concept and thus presents a stepping stone for future work. Therefore, there are numerous possibilities for future investigations.

Firstly, rethinking the current two-dimensional methodology and implementation into a higher-dimensional compatible framework is necessary to align it with clinical practice. This step involves, among others, computational hurdles that need to be addressed. Solutions might include GPU processing and or more efficient Monte Carlo methodologies, in addition to more flexible visualization approaches for the pipeline.

Another direction for future work could be to improve the uncertainty model by researching and implementing a model that supports the complete frequency spectrum of noise (uncertainty). As the addition of medium-frequency noise involves noise patterns, it would be desirable to research and support such. The work of Peischl et al. [PR25] would suit a combination for a more holistic analysis of uncertainty, as they are analyzing medium-frequency noise acquisition artifacts. Consequently, it would be beneficial to follow an established uncertainty classification system (see Chapter 2), which differentiates between what uncertainty categories are viable at which stage of the pipeline, and thus base the uncertainty model on those assumptions. Thus, a user would be able to introduce frequency-flexible uncertainties and the relevant points in the pipeline. Together with a more exhaustive set of available modules and a pipeline that supports workflows as branching or merging, the framework becomes more versatile for testing realistic and relevant scenarios.

Another starting point for future work would be to improve the robustness of the metrics and metric visualizations of the framework. The results show that the viability of the currently used metrics depends on the context. Thus, it may be promising to investigate the matter and find other, possibly more robust, generalizable metrics. In that regard, the framework would offer more metrics and could let the user decide on what metrics to compute and how to visualize them.

The provenance graph offers a lot of opportunities for further work. First, other types of provenance visualizations could be tested for applicability. The current provenance visualization could be improved by adding other graph layouts or by implementing node collapsing strategies to present the graph more neatly, as its size can quickly get out of hand. Moreover, it would be an idea to integrate the uncertainty metrics into the graph itself, for example, by color-coding the nodes. In that way, a user could select an uncertainty of interest and see at first glance which changes on the pipeline had the most impact, and thus, could investigate those in more detail.

All in all, our methodology seems to present a good starting point for the analysis of uncertainty propagation. With further work, more relevant insights into the behavior of uncertainties can be gathered and thus increasing the awareness of uncertainty in health care.

Overview of Generative AI Tools Used

LanguageTool [NM] was used for grammar and spell checking.

Grammarly [SLL] was used for grammar and spell checking.

ChatGPT [Ope] was used for translations, rephrasing, grammar, and spell checking.

Claude [Ant] was used for pseudo code syntax correction.

List of Figures

List of Algorithms

Acronyms

AVOCADO Adaptive Visualization of Comprehensive Analytical Data Origins. 24, 25, 108

CLT Central Limit Theorem. 15, 16, 40, 107

CT Computed Tomography. 7, 12, 16, 22, 66, 107

DTI Diffusion tensor imaging. 13

GP Gaussian Process. 16, 17, 107

GPs Gaussian Processes. 16, 36, 59

i.i.d. independent and identically distributed. 14, 15

LUT Lookup Table. 33, 45, 51, 56, 60, 77, 108, 111

MRI Magnetic Resonance Imaging. 7, 12, 13

PU Parameter Uncertainty. 29, 35–37, 43, 49, 50, 55, 74, 79, 81, 92, 94, 108, 109, 111, 113

PUs Parameter Uncertainties. 30, 31, 37, 40, 54–56, 99, 108, 110

QRI Qualitative Result Inspection. 65

SNR signal-to-noise ratio. 7

VU Visual Uncertainty. 29, 34–37, 43, 44, 49, 50, 56, 68, 69, 72, 78, 88, 108, 109

VUs Visual Uncertainties. 30, 31, 37, 40, 43, 44, 54, 56, 57, 78, 82–84, 86–88, 99, 108–112

W3C World Wide Web Consortium. 22

Bibliography

[AKM+18] Lorenzo Amabili, Jiri Kosinka, Maarten A. J. van Meersbergen, Peter M. A. van Ooijen, Jos B. T. M. Roerdink, Pjotr Svetachov, and Lingyun Yu. *Improving Provenance Data Interaction for Visual Storytelling in Medical Imaging Data Exploration*. The Eurographics Association, 2018.

[Ant] Anthropic. Claude. https://claude.ai [accessed on 07/06/2025].

[BAL12] Ken Brodlie, Rodolfo Allendes Osorio, and Adriano Lopes. A Review of Uncertainty in Data Visualization. In John Dill, Rae Earnshaw, David Kasik, John Vince, and Pak Chung Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. Springer, London, 2012.

[BCS16] Leilani Battle, Remco Chang, and Michael Stonebraker. Dynamic Prefetching of Data Tiles for Interactive Visualization. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 1363–1375, New York, NY, USA, June 2016. Association for Computing Machinery.

[BHJ+14] Georges-Pierre Bonneau, Hans-Christian Hege, Chris R. Johnson, Manuel M. Oliveira, Kristin Potter, Penny Rheingans, and Thomas Schultz. Overview and State-of-the-Art of Uncertainty Visualization. In Charles D. Hansen, Min Chen, Christopher R. Johnson, Arie E. Kaufman, and Hans Hagen, editors, *Scientific Visualization*, pages 3–27. Springer London, London, 2014. Series Title: Mathematics and Visualization.

[BHP] Cynthia Brewer, Mark Harrower, and Pennsylvania State University. ColorBrewer: Color Advice for Maps. https://colorbrewer2.org [accessed on 07/06/2025].

[BJK+16] Tanja Blascheck, Markus John, Kuno Kurzhals, Steffen Koch, and Thomas Ertl. VA2: A Visual Analytics Approach for Evaluating Visual Analytics Applications. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):61–70, January 2016.

[BP16] Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, February 2016.

[Bra00] G. Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 2000. tex.citeulike-article-id: 2236121 tex.posted-at: 2008-01-15 19:21:54 tex.priority: 4.

[BWD+19] Christian Bors, John Wenskovitch, Michelle Dowling, Simon Attfield, Leilani Battle, Alex Endert, Olga Kulyk, and Robert S. Laramee. A Provenance Task Abstraction Framework. *IEEE Computer Graphics and Applications*, 39(6):46–60, November 2019.

[BWR24] Hannah Clara Bayat, Manuela Waldner, and Renata G. Raidou. A Workflow to Visually Assessa Interobserver Variability in Medical Image Segmentation. *IEEE Computer Graphics and Applications*, 44(1):86–94, January 2024. Conference Name: IEEE Computer Graphics and Applications.

[Cam25] Cambridge Dictionary. provenance, May 2025. https://dictionary.cambridge.org/dictionary/english/provenance [accessed on 07/06/2025].

[Co14] Campagnola, Luke and others. PyQtGraph: Scientific graphics and GUI library for python, 2014. https://www.pyqtgraph.org [accessed on 11/07/2025].

[Dev25] jsonpickle Developers. jsonpickle: Python library for serialization of complex objects to JSON, 2025. [accessed on 11/07/2025] https://github.com/jsonpickle/jsonpickle.

[Dur19] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[EFN12] Alex Endert, Patrick Fiaux, and Chris North. Semantic Interaction for Sensemaking: Inferring Analytical Reasoning for Model Steering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2879–2888, December 2012.

[FEM+22] Bernhard Fröhler, Tim Elberfeld, Torsten Möller, Hans-Christian Hege, Julia Maurer, and Christoph Heinzl. Sensitive vPSA – Exploring Sensitivity in Visual Parameter Space Analysis, April 2022. arXiv:2204.01823 [cs].

[GM13] Paul Groth and Luc Moreau. PROV-Overview. An Overview of the PROV Family of Documents, April 2013. Publisher: World Wide Web Consortium http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/ [accessed on 03/05/2025].

[GSo06] Henning Griethe, Heidrun Schumann, and others. The Visualization of Uncertain Data: Methods and Problems. In *SimVis*, volume 2006, January 2006.

[GSWS21] Christina Gillmann, Dorothee Saur, Thomas Wischgoll, and Gerik Scheuermann. Uncertainty-aware Visualization in Medical Imaging - A Survey. *Computer Graphics Forum*, 40(3):665–689, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14333.

[Hab90] Robert Haber. Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in Scientific Computing*, January 1990.

[HB03] Mark Harrower, , and Cynthia A. Brewer. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *The Cartographic Journal*, 40(1):27–37, June 2003. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1179/000870403235002042.

[HG93] G. J. Hunter and M. F. Goodchild. Managing uncertainty in spatial databases: putting theory into practice. *URISA Journal*, 5(2):55–62, January 1993.

[HG15] Rinke Hoekstra and Paul Groth. PROV-O-Viz - Understanding the Role of Activities in Provenance. In Bertram Ludäscher and Beth Plale, editors, *Provenance and Annotation of Data and Processes*, Lecture Notes in Computer Science, pages 215–220, Cham, 2015. Springer International Publishing.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. Publisher: Springer Science and Business Media LLC.

[HSB+22] David Hägele, Christoph Schulz, Cedric Beschle, Hannah Booth, Miriam Butt, Andrea Barth, Oliver Deussen, and Daniel Weiskopf. Uncertainty visualization: Fundamentals and recent developments. *it - Information Technology*, 64(4-5):121–132, August 2022. Publisher: De Gruyter Oldenbourg.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th python in science conference (SciPy 2008)*, pages 11–15. SciPy, 2008.

[Hun07] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. Publisher: IEEE COMPUTER SOC.

[IIJ+13] Tobias Isenberg, Petra Isenberg, Jian Chen, Michael Sedlmair, and Torsten Moller. A Systematic Review on the Practice of Evaluating Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2818–2827, December 2013.

[Jet24] JetBrains s.r.o. PyCharm: Python IDE for professional developers (professional edition), December 2024. https://www.jetbrains.com/pycharm/ [accessed on 11/07/2025].

[KD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31(2):105–112, March 2009.

[KDJ+21] Aasim Kamal, Parashar Dhakal, Ahmad Y. Javaid, Vijay K. Devabhaktuni, Devinder Kaur, Jack Zaientz, and Robert Marinier. Recent advances and challenges in uncertainty visualization: a survey. *Journal of Visualization*, 24(5):861–890, October 2021.

[KG21] Matthias Katzfuss and Joseph Guinness. A General Framework for Vecchia Approximations of Gaussian Processes. *Statistical Science*, 36(1):124–141, 2021. Publisher: Institute of Mathematical Statistics.

[KOF+16] Troy Kohwalter, Thiago Oliveira, Juliana Freire, Esteban Clua, and Leonardo Murta. Prov Viewer: A Graph-Based Visualization Tool for Interactive Exploration of Provenance Data. In Marta Mattoso and Boris Glavic, editors, *Provenance and Annotation of Data and Processes*, Lecture Notes in Computer Science, pages 71–82, Cham, 2016. Springer International Publishing.

[KRKP+16] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and power in academic publishing: Players, agents and agendas*, pages 87 – 90. IOS Press, 2016.

[LLPY07] Claes Lundström, Patric Ljung, Anders Persson, and Anders Ynnerman. Uncertainty Visualization in Medical Volume Rendering Using Probabilistic Animation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1648–1655, November 2007. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[MCER25] M. Musleh, D. Ceneda, H. Ehlers, and R. G. Raidou. ConAn: Measuring and Evaluating User Confidence in Visual Data Analysis Under Uncertainty. *Computer Graphics Forum*, 44(1):e15272, 2025. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.15272.

[Mer23] Neri Merhav. Some Families of Jensen-like Inequalities with Application to Information Theory. *Entropy*, 25(5):752, May 2023. Publisher: Multidisciplinary Digital Publishing Institute.

[Mer25] Merriam-Webster. Definition of PROVENANCE, May 2025. https://www.merriam-webster.com/dictionary/provenance [accessed on 07/06/2025].

[Mic] Microsoft Corporation. Visual Studio Code. https://code.visualstudio.com/ [accessed on 11/07/2025].

[MMT+23] Maath Musleh, Ludvig Paul Muren, Laura Toussaint, Anne Vestergaard, Eduard Gröller, and Renata G. Raidou. Uncertainty guidance in proton therapy planning visualization. *Computers & Graphics*, 111:166–179, April 2023.

[Mor13] Kenneth Moreland. A Survey of Visualization Pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):367–378, March 2013.

[MRO+12] Alan M. MacEachren, Robert E. Roth, James O'Brien, Bonan Li, Derek Swingley, and Mark Gahegan. Visual Semiotics & Uncertainty Visualization: An Empirical Study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2496–2505, December 2012.

[Msm+24] Darcy Mason, scaramallion, mrbean-bremen, rhaxton, Jonathan Suever, Dimitri Papadopoulos Orfanos, Vanessasaurus, Guillaume Lemaitre, Aditya Panchal, Alex Rothberg, Markus D. Herrmann, Joan Massich, James Kerns, Korijn van Golen, Chris Bridge, Simon Biggs, Thomas Robitaille, moloney, Matthew Shun-Shin, Blair Conrad, pawelzajdel, Markus Mattes, YoungKi Lyu, Tim Cogan, Zvi Baratz, Félix C. Morency, Taylor, and Thom Sentner. pydicom/pydicom: pydicom 3.0.1, September 2024. https://github.com/pydicom/pydicom [accessed on 11/07/2025].

[NM] Daniel Naber and Marcin Miłkowski. LanguageTool. https://languagetool.org/ [accessed on 03/05/2025].

[OLM+24] Timbwaoga A. J. Ouermi, Jixian Li, Zachary Morrow, Bart Van Bloemen Waanders, and Chris R. Johnson. Glyph-Based Uncertainty Visualization and Analysis of Time-Varying Vector Fields. In *2024 IEEE Workshop on Uncertainty Visualization: Applications, Techniques, Software, and Decision Frameworks*, pages 73–77, October 2024.

[Ope] OpenAI. ChatGPT. https://chatgpt.com [accessed on 03/05/2025].

[Ots79] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, January 1979.

[PA23] Frederik Hagsholm Pedersen, Jørgensen , Jakob Sauer, , and Martin Skovgaard Andersen. A Bayesian approach to CT reconstruction with uncertain geometry. *Applied Mathematics in Science and Engineering*, 31(1):2166041, December 2023. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/27690911.2023.2166041.

[PB14a] Bernhard Preim and Charl Botha. Chapter 1 - Introduction. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 1–11. Morgan Kaufmann, Boston, January 2014.

[PB14b] Bernhard Preim and Charl Botha. Chapter 10 - Labeling and Measurements in Medical Visualization. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 369–398. Morgan Kaufmann, Boston, January 2014.

[PB14c] Bernhard Preim and Charl Botha. Chapter 2 - Acquisition of Medical Image Data. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 15–67. Morgan Kaufmann, Boston, January 2014.

[PB14d] Bernhard Preim and Charl Botha. Chapter 4 - Image Analysis for Medical Visualization. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 111–175. Morgan Kaufmann, Boston, January 2014.

[PB14e] Bernhard Preim and Charl Botha. Chapter 6 - Surface Rendering. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 229–267. Morgan Kaufmann, Boston, January 2014.

[PB14f] Bernhard Preim and Charl Botha. Chapter 7 - Direct Volume Visualization. In Bernhard Preim and Charl Botha, editors, *Visual Computing for Medicine (Second Edition)*, pages 269–287. Morgan Kaufmann, Boston, January 2014.

[PR25] Jakob Peischl and Renata Raidou. Investigating the Propagation of CT Acquisition Artifacts along the Medical Imaging Pipeline:. In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 752–764, Porto, Portugal, 2025. SCITEPRESS - Science and Technology Publications.

[PRJ12] Kristin Potter, Paul Rosen, and Chris R. Johnson. From Quantification to Visualization: A Taxonomy of Uncertainty Visualization Approaches. In Andrew M. Dienstfrey and Ronald F. Boisvert, editors, *Uncertainty Quantification in Scientific Computing*, IFIP Advances in Information and Communication Technology, pages 226–249, Berlin, Heidelberg, 2012. Springer.

[PWL97a] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.

[PWL97b] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.

[Pyt23] Python Software Foundation. Python: Programming language, 2023. https://www.python.org/ [accessed on 11/07/2025].

[Rai18] Renata G. Raidou. *Uncertainty Visualization: Recent Developments and Future Challenges in Prostate Cancer Radiotherapy Planning*. The Eurographics Association, 2018.

[Ras04] Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, pages 63–71. Springer, Berlin, Heidelberg, 2004.

[RHM+22] Patrick M. Reed, Antonia Hadjimichael, Keyvan Malek, Tina Karimi, Chris R. Vernon, Vivek Srikrishnan, Rohini S. Gupta, David F. Gold, Ben Lee, Klaus Keller, Travis B. Thurber, and Jennie S. Rice. *Addressing uncertainty in multisector dynamics research*. Zenodo, 2022.

[Rod25] Giampaolo Rodola. psutil: process and system utilities for Python, 2025. https://github.com/giampaolo/psutil [accessed on 11/07/2025].

[RPHL14] Gordan Ristovski, Tobias Preusser, Horst K. Hahn, and Lars Linsen. Uncertainty in medical visualization: Towards a taxonomy. *Computers & Graphics*, 39:60–73, April 2014.

[SAB+19] Andrea Saltelli, Ksenia Aleksankina, William Becker, Pamela Fennell, Federico Ferretti, Niels Holst, Sushan Li, and Qiongli Wu. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. *Environmental Modelling & Software*, 114:29–39, April 2019.

[Sal02] Andrea Saltelli. Sensitivity Analysis for Importance Assessment. *Risk Analysis*, 22(3):579–590, 2002. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00040.

[SBO+23] R. Schaer, M. Bach, M. Obmann, K. Flouris, E. Konukoglu, B. Stieltjes, H. Müller, C. Aberle, O.A. Jimenez del Toro, and A. Depeursinge. Task-Based Anthropomorphic CT Phantom for Radiomics Stability and Discriminatory Power Analyses (CT-Phantom4Radiomics) [Data set]., 2023. tex.howpublished: Data set, The Cancer Imaging Archive.

[SGB+24] Shanu Saklani, Chitwan Goel, Shrey Bansal, Zhe Wang, Soumya Dutta, Tushar M. Athawale, David Pugmire, and Christopher R. Johnson. Uncertainty-Informed Volume Visualization using Implicit Neural Representation. In *2024 IEEE Workshop on Uncertainty Visualization: Applications, Techniques, Software, and Decision Frameworks*, pages 62–72, October 2024.

[SLL] Alex Shevchenko, Max Lytvyn, and Dmytro Lider. Grammarly. https://grammarly.com [accessed on 07/06/2025].

[SLSG16] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg. AVOCADO: Visualization of Workflow–Derived Data Provenance for Reproducible Biomedical Research. *Computer Graphics Forum*, 35(3):481–490, 2016. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12924.

[SLSR10] Meredith Skeels, Bongshin Lee, Greg Smith, and George G. Robertson. Revealing Uncertainty for Information Visualization. *Information Visualization*, 9(1):70–81, January 2010. Publisher: SAGE Publications.

[SMH10] Ahmed Saad, Torsten Möller, and Ghassan Hamarneh. ProbExplorer: Uncertainty-guided Exploration and Editing of Probabilistic Medical Image Segmentation. *Computer Graphics Forum*, 29(3):1113–1122, 2010. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01691.x.

[SS17] Andreas Schreiber and Regina Struminski. Visualizing Provenance using Comics. In *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017)*, Seattle, WA, 2017. USENIX Association.

[Thea] The Cancer Imaging Archive (TCIA). Welcome to The Cancer Imaging Archive. https://www.cancerimagingarchive.net/ [accessed on 03/05/2025].

[Theb] The Qt Company. Qt: The Future of Digital Experiences. https://www.qt.io [accessed on 11/07/2025].

[The25] The Qt Company. PySide6: Qt for python, 2025. https://doc.qt.io/qtforpython/ [accessed on 11/07/2025].

[THM+05] Judi Thomson, Elizabeth Hetzler, Alan MacEachren, Mark Gahegan, and Misha Pavel. A typology for visualizing uncertainty. In *Visualization and Data Analysis 2005*, volume 5669, pages 146–157. SPIE, March 2005.

[Tom] Tom Schimansky. CustomTkinter: Modern and customizable python UI-library based on tkinter. https://customtkinter.tomschimansky.com/ [accessed on 11/07/2025].

[Uch13] Seiichi Uchida. Image processing and recognition for biological images. *Development, Growth & Differentiation*, 55(4):523–549, May 2013. Publisher: John Wiley & Sons, Ltd.

[Vec88] A. V. Vecchia. Estimation and Model Identification for Continuous Spatial Processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):297–312, January 1988.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

[VVWS23] Christiaan G. A. Viviers, M. M. Amaan Valiuddin, Peter H. N. de With, and Fons van der Sommen. Probabilistic 3D segmentation for aleatoraic uncertainty quantification in full 3D medical data. In *Medical Imaging 2023: Computer-Aided Diagnosis*, volume 12465, pages 341–351. SPIE, April 2023.

[Wei22] Daniel Weiskopf. Uncertainty Visualization: Concepts, Methods, and Applications in Biological Data Visualization. *Frontiers in Bioinformatics*, 2, 2022.

[Wel62] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, August 1962.

[Wor13a] World Wide Web Consortium. PROV-DM: The PROV Data Model, 2013. https://www.w3.org/TR/2013/REC-prov-dm-20130430/ [accessed on 03/05/2025].

[Wor13b] World Wide Web Consortium. PROV Model Primer, 2013. https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/ [accessed on 03/05/2025].

[Wor13c] World Wide Web Consortium. PROV-Overview, 2013. https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/ [accessed on 03/05/2025].

[WSNI+14] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, June 2014. Publisher: PeerJ Inc.

[XOW+20] Kai Xu, Alvitta Ottley, Conny Walchshofer, Marc Streit, Remco Chang, and John Wenskovitch. Survey on the Analysis of User Interactions and Visualization Provenance. *Computer Graphics Forum*, 39(3):757–783, 2020. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14035.

[Zha21] Jiaxin Zhang. Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey. *WIREs Computational Statistics*, 13(5):e1539, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1539.

[ZLL+21] Binsheng Zhao, Qin Li, Yongguang Liang, Hao Yang, Marios A. Gavrielides, Lawrence H. Schwartz, Daniel C. Sullivan, and Nicholas A. Petrick. QIBA anthropomorphic abdominal phantom CT scans, 2021.